UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

TUTORIAL INTERACTIVO EN ANDROID SOBRE CIRCUITOS ARITMÉTICOS

Mariano Jose Casado Abril
Tutor: Eduardo Boemo Scalvinoni

Enero 2017

TUTORIAL INTERACTIVO EN ANDROID SOBRE CIRCUITOS ARITMÉTICOS

AUTOR: Mariano Jose Casado Abril
TUTOR: Eduardo Boemo Scalvinoni

Digital System Laboratory
Dpto. Tecnología Electrónica y de Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero de 2017

Resumen

Este Trabajo Fin de Grado tiene como objetivo el diseño y desarrollo de una aplicación móvil para el sistema operativo Android. Dicha aplicación servirá para desarrollar una guía de problemas sobre Circuitos Aritméticos para la asignatura “Circuitos Electrónicos Digitales” de primer curso del Grado de Ingeniería de Tecnologías y Servicios de Telecomunicación.

Los contenidos incluidos son adaptaciones de los enunciados disponibles en la actual guía de problemas impresa. Se han seleccionado aquellos aptos para su desarrollo en forma de ejercicios interactivos.

Se ha incluido una selección de diapositivas diseñadas por el profesor de la asignatura en forma de tutorial, que contiene los conceptos que el alumno debe conocer sobre los Circuitos Aritméticos.

A través del desarrollo de este proyecto se pretende facilitar la realización y aprendizaje de los ejercicios de la asignatura aprovechando las ventajas que los dispositivos móviles ofrecen a los alumnos.

El resultado estará disponible para su descarga gratuita en la plataforma Google Play Store.

Abstract

This End-of-Grade Project aims to design and develop a mobile application for the Android operating system. This application will serve to develop a problem guide on Arithmetic Circuits for the subject "Digital Electronic Circuits" of the first year of the Engineering Degree in Telecommunication Technologies and Services.

The contents included are adaptations of the statements available in the current print problem guide. They have been selected for their development as interactive exercises.

A selection of slides designed by the teacher of the subject has been included in the form of a tutorial, which contains the concepts that the student should know about Arithmetic Circuits.

Through the development of this project we want to facilitate the learning of the exercises of the subject taking advantage of the features that mobile devices offer to the students.

The result will be available for free download on the Google Play Store platform.

Palabras clave

Android, aplicación, móvil, circuitos, aritméticos, electrónica, digital, divisor, memoria, hexadecimal, binario, multiplicador, decimal, formatos, números, tutorial.

Keywords

Android, application, mobile, circuits, arithmetic, electronics, digital, dividing, memory, hexadecimal, binary, multiplier, decimal, formats, ,numbers, tutorial.

Agradecimientos

Me gustaría agradecer a mis padres lo mucho que han aguantado durante estos años de carrera.

A Eduardo Boemo por haberme permitido desarrollar este proyecto.

A mis amigos por soportar todas las veces en las que les he enseñado como progresaba la aplicación.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

En los últimos años hemos podido comprobar como se ha producido una notable evolución en cuanto a lo que a dispositivos móviles se refiere. Hemos pasado de la utilización de simples teléfonos móviles a la aparición tanto de tabletas como terminales inteligentes, también conocidos como "smartphones". Esta nueva tecnología ha alcanzado un nivel de uso tan elevado en la sociedad actual, que prácticamente cualquier persona dispone al menos de un dispositivo de este tipo.

Este proyecto parte de la idea de poder aprovechar las ventajas que esta nueva tecnología ofrece para poder facilitar tanto enseñanza, como aprendizaje de los alumnos, siguiendo la línea de las anteriores aplicaciones móviles desarrolladas por DSLab.

Con este proyecto se podrá lograr sustituir las tradicionales guías de problemas de la asignatura de Circuitos Electrónicos Digitales, distribuidas a través de archivos PDF o impresas, por una aplicación que las contenga.

Se ha desarrollado una aplicación capaz de evaluar los conceptos relacionados con los circuitos aritméticos, capaz de mostrar un tutorial, que permita a los alumnos obtener una evaluación inmediata de sus respuestas y permita enviar sus resultados tanto a otros alumnos como al profesor.

Podemos encontrar las siguientes ventajas en el uso de este tipo de dispositivos inteligentes:

- Fácil uso.
- Eliminación del formato físico.
- Utilización sin necesidad de conexión a Internet.
- Facilita el estudio en cualquier lugar.
- Corrección instantánea de errores.
- Posibilidad de comparar y compartir resultados con otros usuarios, tanto con otros alumnos, como con el profesor.
- Utilización en espacios con poca luz.

1.2 Objetivos

El objetivo de este Trabajo de Fin de Grado es el desarrollo de una herramienta, en forma de aplicación móvil gratuita, dirigida a los alumnos de la asignatura Circuitos Electrónicos Digitales de la EPS.

Esta aplicación ofrece a los alumnos un tutorial en forma de diapositivas donde están contenidos los conceptos básicos del tema a tratar y una serie de problemas interactivos que los alumnos podrán resolver gracias a los contenidos aportados en el tutorial.

La idea de este proyecto surge del análisis y aprendizaje del funcionamiento de los circuitos aritméticos. Los problemas que aparecerán en la aplicación tendrán cada uno un grado de complejidad diferente.

El usuario tendrá la posibilidad de comprobar instantáneamente su nivel de conocimiento, e incluso podrá guardar y enviar sus soluciones, tanto para compartirlas con los otros alumnos como con el profesor, para resolver las posibles dudas que puedan surgir sobre la resolución de los ejercicios.

El resultado de este proyecto sirve como complemento a la guía de problemas convencional de la asignatura, indicando en ella que ejercicios son los que aparecen en la aplicación y se pueden realizar de manera interactiva.

Los conceptos relacionados con los Circuitos Aritméticos que se desean plasmar en este proyecto son los siguientes:

- Formatos de números.
- Extensión de signo.
- Suma.
- Multiplicación binaria.
- División binaria.
- Comparación de números binarios.
- Tablas de Look-Up.

1.3 Organización de la memoria

Este documento tiene como objetivo mostrar el desarrollo del Trabajo de Fin de Grado sobre una aplicación para Android para evaluar los conocimientos sobre Circuitos Aritméticos. Para ello se seguirá la siguiente estructura:

- Capítulo 1: Motivación. Objetivos. Organización de la memoria.
- Capítulo 2: Estado del arte. Aplicaciones similares.
- Capítulo 3: Diseño. Funcionalidad, limitaciones.
- Capítulo 4: Desarrollo.
- Capítulo 5: Integración, pruebas y resultados.
- Capítulo 6: Conclusiones y trabajo futuro.

2 Estado del arte

Como se ha mencionado en el Apartado 1.1 de este proyecto, el objetivo de diseño del mismo es la creación de una aplicación a través de la cual los alumnos puedan aumentar sus conocimientos relativos a conceptos relacionados con los circuitos aritméticos.

Por ello se ha decidido realizar un estudio sobre posibles aplicaciones similares que haya disponibles para su descarga.

Las aplicaciones más similares al concepto que se desea desarrollar son las siguientes:

2.1 Binary Arithmetic

Esta aplicación disponible en la web https://apkpure.com/ permite a los usuarios realizar algunas operaciones binarias tales como: suma, resta, multiplicación, división, XOR o AND.

El defecto que se puede encontrar en ella con respecto a la que se ha desarrollado es que simplemente realiza las operaciones deseadas, pero en ningún momento ayuda a comprender como se realizan.

2.2 Digital Electronics

Esta aplicación disponible de manera gratuita en Google Play Store nos ofrece un libro completo sobre Electrónica Digital y circuitos, cubriendo los temas importantes acerca de la materia, incluyendo notas, materiales, noticias o un blog.

Dispone un formato similar a la idea planteada en la aplicación que se ha desarrollado en el proyecto a modo de tutorial.

El aspecto interesante de esta aplicación es que incluye una gran variedad de temas, pero su problema es que no incluye ningún tipo de ejercicio para comprobar los conocimientos obtenidos.

2.3 Digital Electronics 101

Esta aplicación disponible de manera gratuita en Google Play Store nos ofrece un test con soluciones del tipo respuesta múltiple, junto a sus soluciones. A su vez incluye un tutorial de teoría con una gran variedad de temas relacionados con la Electrónica Digital.

Sus contenidos son los más completos que se han encontrado, ya que permite a los usuarios comprender una gran cantidad de aspectos relacionados con Electrónica Digital, y posteriormente aplicarlos en un test con solución.

Tras su análisis, se observa que la idea planteada en esta aplicación es la que más se asemeja a la idea de la que parte este proyecto.

2.4 Digital Electronics 2

Esta aplicación disponible de manera gratuita en Google Play Store nos ofrece un conjunto de anotaciones simples a través de las cuales trata los conceptos más importantes relacionados con los siguientes temas:

- Sistemas numéricos y códigos binarios.
- Puertas universales.
- Multivibradores.
- Memorias semiconductoras.
- Familias lógicas.

La idea que plantea es interesante, ya que aborda dos de los temas que estarán incluidos en la aplicación que se ha desarrollado, los sistemas y códigos binarios, y las memorias.

En ella se echa en falta la inclusión de un test o algún tipo de ejercicio en los que podamos evaluar los conocimientos obtenidos.

Una vez se han estudiado las aplicaciones similares encontradas por la web podemos concluir con que no hay ninguna disponible para descarga que incluya todas las características incluidas en la que se ha desarrollado ya que, por ejemplo, ninguna ofrece una serie de ejercicios interactivos excepto la primera. Lo más similar son ejercicios tipo test.

Como ya se ha mencionado en la introducción nuestra aplicación cuenta con un sistema de envío de soluciones. Esta característica no está disponible en ninguna de las aplicaciones encontradas.

En la tabla que se muestra a continuación se comparan las aplicaciones previamente analizadas con la que se ha desarrollado a través de este proyecto.

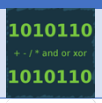
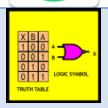
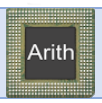
APP	Tutorial	Ejercicios	Test	Enviar Ejercicios	Icono
Binary Arithmetic	No	Si	No	No	
Digital Electronics	Si	No	No	No	
Digital Electronics 101	Si	No	Si	No	
Digital Electronics 2	Si	No	No	No	
Arithmetic Circuits	Si	Si	No	Si	

Tabla 2-1: Tabla comparativa de aplicaciones similares

3 Diseño

3.1 Objetivos de diseño

El objetivo de este proyecto es el diseño de una aplicación para teléfonos móviles a través de la cual se puedan evaluar los conocimientos sobre los Circuitos Aritméticos.

Su propósito principal es solucionar las dudas que tengan los alumnos de la asignatura, pero puede ser utilizada por cualquier persona que tenga interés en el aprendizaje sobre este tema.

Cuenta con un tutorial de diapositivas y una lista de 18 ejercicios interactivos sobre los Circuitos Aritméticos. Se pueden comprobar, guardar y enviar las soluciones de estos ejercicios.

3.2 Requisitos de diseño

3.2.1 Sistemas operativos compatibles

Uno de los problemas que surgen a la hora de comenzar a desarrollar una aplicación para dispositivos móviles es la elección de los sistemas operativos para los que estará disponible. Este es un problema importante ya que, no solo a nivel técnico, sino también a nivel de diseño, nuestra aplicación estará relacionada con el sistema operativo en el que vaya a funcionar.

En nuestro caso el sistema operativo elegido para el desarrollo de la aplicación es Android debido a diversas razones que se expondrán a continuación.

El primer y principal motivo es el relacionado con el software de desarrollo de la aplicación. Para desarrollar nuestro proyecto se ha utilizado Android Studio, un software gratuito y multiplataforma. En el caso de iOS, el software necesario para el desarrollo de las aplicaciones solo puede ser utilizado en dispositivos Apple.

Otro punto importante a la hora de la elección de Android como sistema operativo es que ofrece una mayor facilidad a la hora de hacer pruebas con las aplicaciones. Éstas se pueden realizar en cualquier dispositivo sin disponer de una licencia de desarrollador. Tiene un precio de 25$ y será necesaria a la hora de publicar la aplicación. En el caso de iOS, si queremos realizar pruebas necesitamos una licencia de desarrollador cuyo precio alcanza los 90$, y solo podemos realizar comprobaciones en un dispositivo.

Aunque inicialmente esta aplicación está dirigida a los alumnos de la asignatura “Circuitos Electrónicos Digitales”, siempre que se desarrolla un proyecto de este tipo se desea llegar al mayor número de usuarios posibles, por ello se ha realizado un estudio sobre la distribución de Android e iOS sobre el total de dispositivos móviles inteligentes que encontramos a día de hoy.

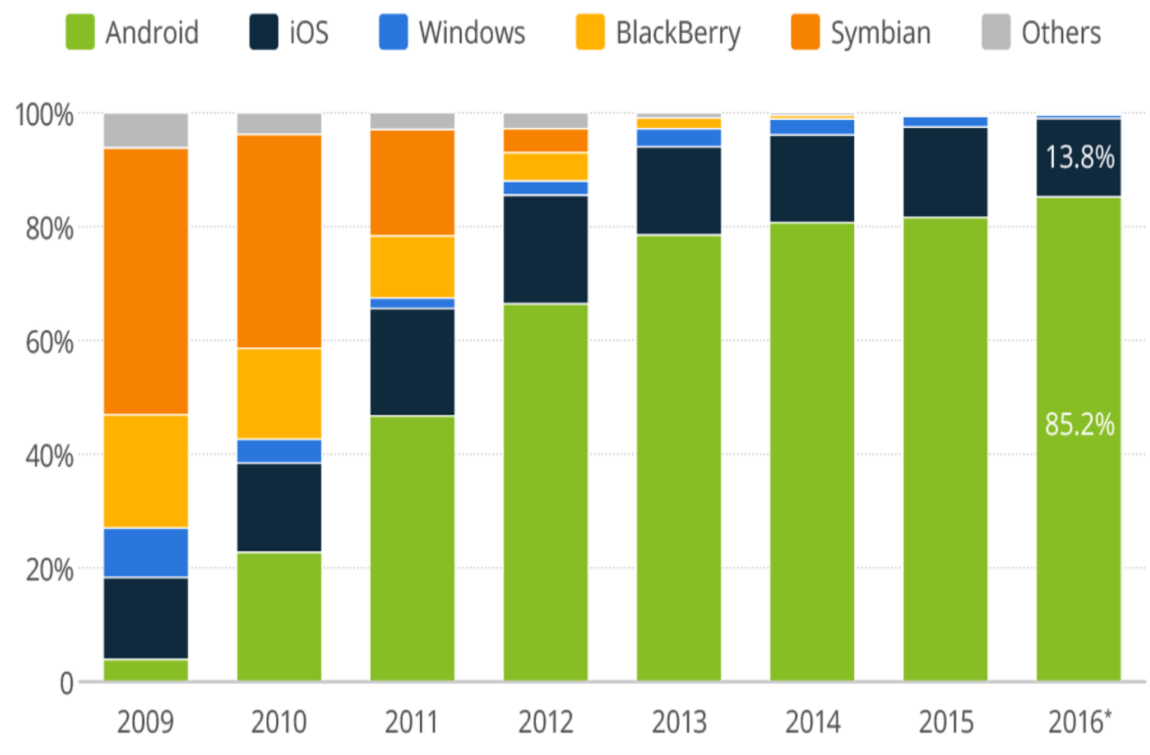


Figura 3-1: Distribución del mercado de "Smartphones" en 2016 (Fuente: Business Insider)

Podemos observar una notable diferencia entre el uso de dispositivos de ambos tipos. Aunque en años anteriores podemos apreciar la existencia de otros sistemas operativos, hoy en día predominan los que funcionan con iOS o Android. Entre estos dos tipos de dispositivos, como hemos mencionado anteriormente, hay una enorme diferencia en lo que a número de dispositivos vendidos se refiere.

Observando que Android ocupa un porcentaje del 85% del mercado, frente al 13.8% de iOS es sencillo comprender nuestra elección.

3.2.1 Versiones compatibles

A continuación se explicarán la elección de las versiones de Android para las cuáles estará disponible nuestro diseño.

El único problema que encontramos en este punto es que si la diseñamos para las últimas versiones de Android, los usuarios que dispongan de versiones anteriores no podrán instalarla. En el caso contrario si la desarrollamos para versiones anteriores, los usuarios de nuevas versiones si podrán instalarla.

A través de los datos proporcionados por la página web https://developer.android.com/ podemos seleccionar las versiones compatibles con facilidad.

Versión	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3-2.3.7	Gingerbread	10	1.7%
4.0.3-4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	Kit Kat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshallow	23	15.2%

Tabla 3-1: Distribución versiones Android en junio de 2016

Se ha decidido diseñar la aplicación para todos aquellos dispositivos que tengan un sistema operativo Android con una versión entre la denominada "*Ice Cream Sandwich*" y la más actual.

3.2.2 Tipos de dispositivos compatibles

Se han descartado tanto los relojes como las televisiones inteligentes ya que no son los dispositivos más adecuados para el uso de una aplicación de carácter docente.

3.2.3 Tipos de pantalla compatibles

Para la selección de los tipos de pantalla que puede soportar nuestra aplicación debemos tener en cuenta que el sistema operativo Android se ejecuta en dispositivos con pantallas con diferentes características. Este sistema operativo es capaz de realizar cambios automáticos de tamaño en la aplicación para que esta se ajuste a cualquier dispositivo, pero realizando un estudio y posterior selección de los tipos de pantalla que deseamos en nuestro diseño, podemos mejorar tanto la experiencia de usuario como el rendimiento.

Podemos clasificar las pantallas de los dispositivos Android según cuatro criterios:

- Tamaño de pantalla.
- Densidad de pantalla.
- Resolución.
- Orientación de la pantalla.

En el diseño de esta aplicación nos centraremos en los criterios de tamaño y densidad de pantalla.

El tamaño de pantalla es el tamaño físico real de la misma. Para su medición se considera la longitud en pulgadas de la diagonal. Según este criterio podemos clasificarlas en cuatro rangos: *small, normal, large, extralarge*.

La densidad de pantalla es la cantidad de píxeles dentro del área física que esta posee. Normalmente se mide en dpi (puntos por pulgada). Según este criterio las podemos clasificar las en 6 rangos: baja, media, alta, extraalta, extra extraalta y extra extra extraalta.

Los tamaños y densidades generalizados se organizan en torno a una configuración de referencia con un tamaño normal y una densidad de pantalla media. Como existe un número infinito de valores de tamaño y densidad de pantalla, Android agrupa los dispositivos en los rangos anteriores, tanto de tamaño como de densidad, de la siguiente manera:

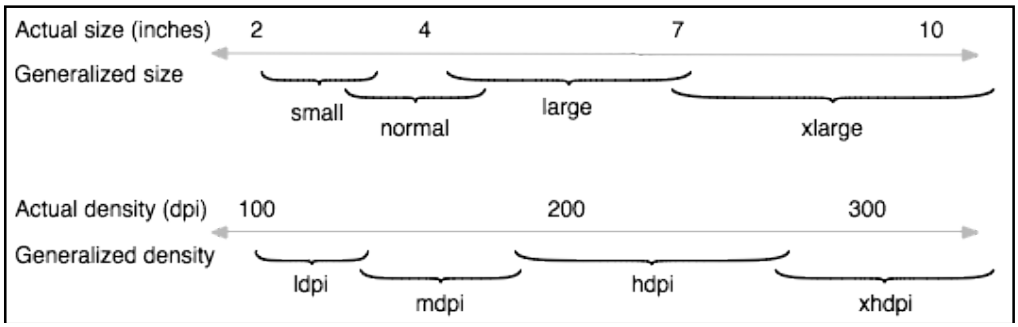


Figura 3-2: Asignación aproximada de tamaños y densidades reales a tamaños y densidades generalizados

Una vez que conocemos como se clasifican los dispositivos según las pantallas podemos seleccionar cuales serán aptos para nuestra aplicación.

En primer lugar se estudiará la elección de los dispositivos válidos según el tamaño de pantalla.

En la tabla mostrada en la siguiente figura observamos los datos de la distribución de dispositivos sobre el total existentes según el tamaño de pantalla con los datos recopilados hasta el día 1 de Agosto de 2016.

Tamaño de pantalla	Total
Small	1.8%
Normal	86.7%
Large	7.6%
Xlarge	3.9%

Tabla 3-2: Distribución dispositivos Android según tamaño de pantalla

Observando la Tabla 3-2 podemos observar como existe un predominio, sobre el conjunto global de los dispositivos Android, de aquellos dotados con pantallas de tamaño normal. El resto de tamaños tienen unos porcentajes de uso muy inferiores.

Debido a ello se ha decidido diseñar la aplicación para dispositivos con tamaño de pantalla normal, e incluir los dispositivos con tamaño de pantalla *large*, ya que viendo antiguos estudios, se observa como el número de estos ha aumentado. En los otros dos casos, tanto *small* como *extralarge,* ha habido una disminución del número de dispositivos sobre el total.

Otro de los motivos por los cuales se ha decidido desarrollar éste proyecto solo para dispositivos con pantalla normal y large es que los dispositivos que tienen tamaño small y extralarge son los conocidos como "*SmartWatches*" y "*SmartTV*". Estos dos tipos de dispositivos no son los más apropiados para utilizar una aplicación con fines docentes, y en

el caso de los relojes inteligentes, su reducido tamaño de pantalla dificultaría mucho el correcto aprovechamiento de las características que ofrece este proyecto.

En segundo lugar se ha estudiado la elección de los dispositivos válidos según la densidad de pantalla.

En la tabla mostrada en la siguiente figura observamos los datos de la distribución de dispositivos sobre el total existentes según la densidad de pantalla con los datos recopilados hasta el día 1 de Agosto de 2016.

Densidad de pantalla	Total
ldpi	2.0%
mdpi	11.0%
tvdpi	2.2%
hdpi	40.8%
xhdpi	28.5%
xxhdpi	15.5%

Tabla 3-3: Distribución dispositivos Android según densidad de pantalla

Observando la tabla superior (3-3) podemos comprobar como existe un predominio sobre el conjunto global de los dispositivos, de aquellos dotados con pantallas con densidades hdpi, xhdpi y xxhdpi. El resto de valores tienen unos porcentajes de uso muy inferiores.

Debido a ello se ha decidido establecer un valor de densidad estándar para todas las figuras que aparezcan en el proyecto, que se encuentre en torno al valor intermedio entre hdpi y xhdpi.

Teniendo en cuenta que la densidad hdpi tiene un valor de 240 dpi y la xhdpi tiene un valor de 320 dpi se diseñarán todas las imágenes con una densidad de 300 dpi.

A continuación para unir las dos elecciones que hemos realizado, tanto de tamaño de pantalla como de densidad, se muestra una tabla en la que se puede observar el porcentaje de uso de los dispositivos Android según su tamaño de pantalla y densidad.

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	1.6%						1.6%
Normal		3.1%	0.2%	38.7%	30.4%	15.5%	89.7%
Large	0.2%	3.9%	1.9%	0.4%	0.4%		6.8%
Xlarge		2.8%		0.3%	0.6%		3.7%
Total	1.8%	9.8%	2.1%	39.4%	31.4%	15.5%	

Tabla 3-4: Distribución dispositivos Android según tamaño y densidad de pantalla

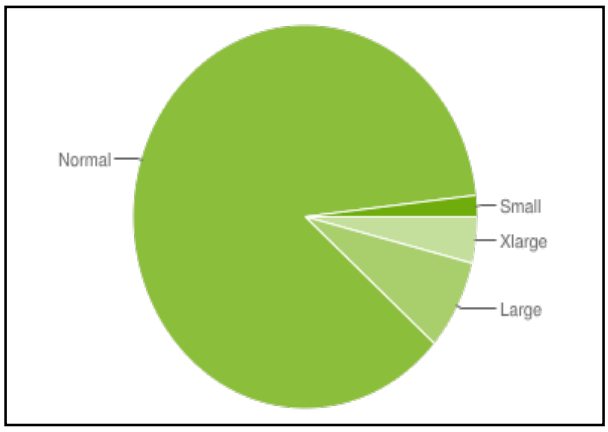


Figura 3-3: Distribución dispositivos Android según tamaño de pantalla

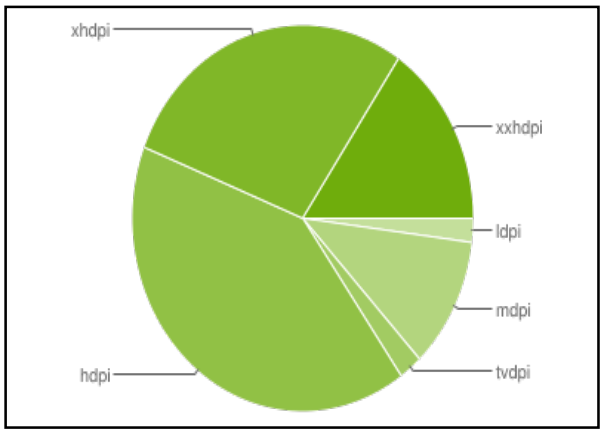


Figura 3-4: Distribución dispositivos Android según densidad de pantalla

Todas las imágenes y tablas de esta sección pertenecen a la página web https://developer.android.com/

3.3 Diagrama de bloques del diseño

Como hemos mencionado anteriormente, el objetivo de este proyecto es el desarrollo de una aplicación móvil con un objetivo docente. Por lo tanto es necesario que nuestro diseño sea lo más simple e intuitivo posible. Para conseguirlo se ha decidido seguir la estructura que siguen las anteriores aplicaciones desarrolladas por DSLab.

Dicha estructura se organizaría en forma de árbol. De esta manera, a través de un menú principal, el usuario será capaz de acceder directamente a todas las funcionalidades que ofrece nuestra aplicación, tanto el tutorial de la asignatura, menú de ayuda, lista de ejercicios o a la opción para enviar los mismos a través del correo electrónico.

En el siguiente diagrama se refleja el diseño en árbol del que hablábamos anteriormente:

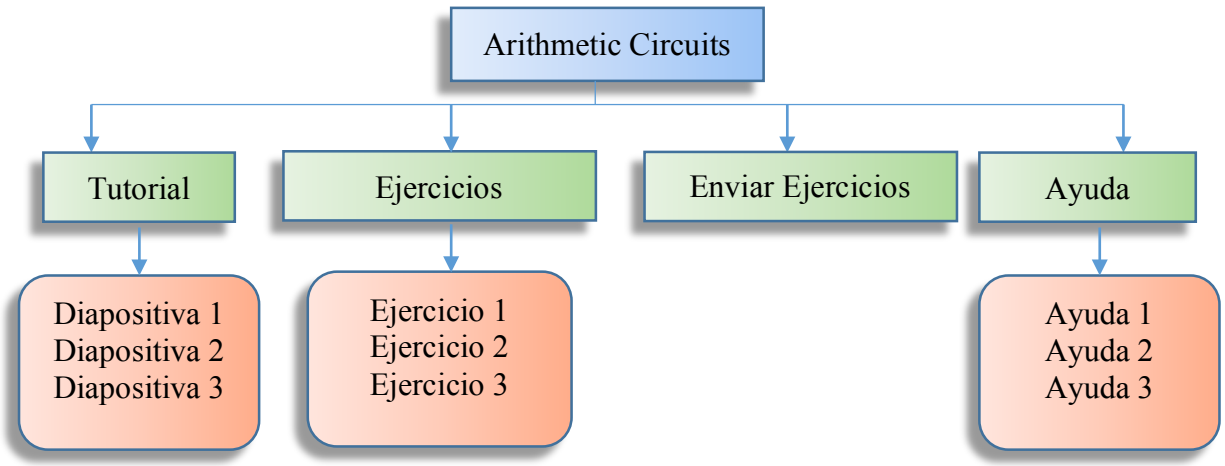


Figura 3-5: Diagrama de bloques de la aplicación

3.4 Estructura de la aplicación

A continuación se mostrarán de una manera resumida los diferentes módulos de la aplicación; es decir, las diferentes pantallas con las que el usuario podrá interactuar.

3.4.1 Menú principal

Desde este menú formado por cuatro botones el usuario puede acceder a todas las funcionalidades disponibles.

- Accederá al tutorial al pulsar el botón con el mismo nombre.
- Accederá a la lista de ejercicios al pulsar el botón "*Exercises*"
- Accederá a la opción de enviar los ejercicios por e-mail al pulsar el botón "*Send Exercises*"
- Accederá al menú de ayuda al pulsar el botón "*Help*".

3.4.2 Tutorial Circuitos Aritméticos

En esta sección el usuario puede acceder a un tutorial mostrado en forma de diapositivas.

Han sido diseñadas por Eduardo Boemo, profesor de la asignatura "Circuitos Electrónicos Digitales" y están basadas en los contenidos de la misma relacionados con los Circuitos Aritméticos.

3.4.3 Ayuda

El usuario dispone de una sección de ayuda, accesible desde todas las diferentes secciones de la aplicación.

En esta sección encontrará una lista de conceptos importantes para comprender como realizar un uso correcto de la aplicación y así solucionar los posibles problemas que puedan surgirle al usuario a la hora de comenzar a utilizarla.

3.4.4 About

Esta sección es la más básica de toda la aplicación y es accesible desde cualquier otra. En ella simplemente se muestra una imagen con todos los datos relativos al desarrollador del proyecto. Este tipo de sección es más conocido como "*Acerca de*".

3.4.5 Guardar ejercicios

El usuario cuenta en cada ejercicio con un botón "*Save*". Al pulsar este botón dispone de un menú emergente en el que se le solicita que introduzca el nombre con el que desea guardar la solución del ejercicio. En el caso de que ese nombre de ejercicio coincida con uno anteriormente guardado, se indica este problema, ofreciendo la opción de sobrescribir una solución anteriormente guardada.

El directorio en el que se guardan las diferentes soluciones del dispositivo se crea en el momento en el que se inicie por primera vez la aplicación en su dispositivo.

3.4.6 Cargar o Borrar Ejercicios

Una vez que el usuario selecciona que ejercicio desea resolver se muestra el enunciado del mismo junto a dos opciones, iniciar la resolución, y cargar o borrar una solución previamente guardada.

En el caso en el que se desee cargar una solución previamente guardada simplemente se debe seleccionar una de las opciones que aparecerán en la lista y pulsar el botón "*Load*". Si se selecciona más de una o ninguna se mostrará un mensaje de error. Si el proceso de carga de la solución es correcto, se muestra un mensaje al usuario indicando que el proceso se ha efectuado con éxito.

En el caso en el que el usuario desee borrar una solución previamente guardada simplemente debe seleccionar cual desea borrar y pulsar el botón "*Delete*", pudiendo seleccionar más de una simultáneamente. Si no se ha seleccionado ninguna se mostrará un mensaje de error. Si el proceso de borrado de la solución es correcto, se muestra un mensaje al usuario indicando el éxito del mismo.

3.4.7 Enviar ejercicios

El usuario puede acceder a esta funcionalidad a través del menú principal pulsando el botón "*Send Exercises*".

Una vez que el usuario pulse este botón aparecerá un menú emergente en el que podrá seleccionar una o más soluciones previamente guardadas. En el momento en el que el usuario realice la selección debe pulsar el botón "*Send exercises*" para acceder a una aplicación de correo electrónico.

3.4.8 Corregir ejercicios

Esta función está disponible en todos los ejercicios. El usuario accede a ella pulsando el botón "*Check*" presente en cualquier pantalla de resolución.

Una vez que el usuario ha pulsado este botón se muestran diversos mensajes en pantalla. Éstos indicarán acierto o fallo según el resultado de la corrección.

3.4.9 Resolver ejercicios

En ciertos ejercicios el usuario dispone de una opción a través de la cual podrá mostrar la solución correcta de un ejercicio.

Esta función es accesible desde el botón "*Solve*" y su función es facilitar el aprendizaje en ejercicios que tengan cierta dificultad.

3.4.10 Resetear ejercicios

Esta función está disponible en todos los ejercicios. El usuario accede a ella pulsando el botón "*Reset*" presente en cualquier pantalla de resolución.

Cuando se pulsa el botón anteriormente mencionado se mostrará un cuadro de diálogo en el que se preguntará si realmente quiere borrar los datos que introducidos en el ejercicio. En caso afirmativo el ejercicio vuelve a sus valores por defecto.

3.4.11 Menú emergente

El usuario dispone de un menú emergente en todas las pantallas anteriormente mencionadas de la aplicación. Este menú se desplegará pulsando el botón de menú de Android. A través del uso de este botón conseguimos la aparición de un menú alternativo, que solo está presente en pantalla en el momento en el que el usuario lo demanda.

El objetivo del diseño de este menú es facilitar el acceso tanto a la Ayuda como al "*Acerca de*" de la aplicación.

3.4.12 Ejercicios

Ejercicios 1 y 3
Concepto: Formatos de números, extensión de signo
Descripción: El usuario tendrá que introducir el valor binario correspondiente a ciertos números decimales utilizando el formato de complemento a dos, con unas dimensiones determinadas según lo requiera el enunciado.

Tabla 3-5: Ejercicios 1 y 3

Ejercicio 2
Concepto: Formatos de números, conversión decimal a binario
Descripción: El usuario tendrá que introducir el valor decimal correspondiente a ciertos números binarios.

Tabla 3-6: Ejercicio 2

Ejercicios 4 y 5
Concepto: Formatos de números, conversión decimal a hexadecimal
Descripción: El usuario tendrá que introducir el valor decimal correspondiente a ciertos números hexadecimales, o el valor hexadecimal correspondiente a ciertos números decimales, según se demande en el enunciado.

Tabla 3-7: Ejercicios 4 y 5

Ejercicios 6 y 7
Concepto: Formatos de números, conversión hexadecimal a binario
Descripción: El usuario tendrá que introducir el valor binario correspondiente a ciertos números hexadecimales, o el valor hexadecimal correspondiente a ciertos números binarios, según se demande en el enunciado.

Tabla 3-8: Ejercicios 6 y 7

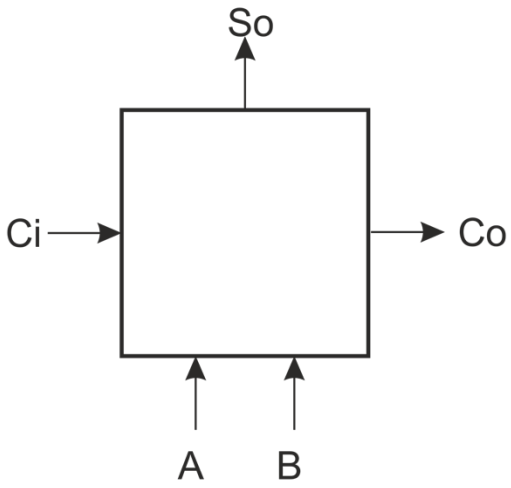
Ejercicio 8
Concepto: Suma
Descripción: El usuario encontrará una figura de un circuito full-adder para comprender mejor el ejercicio. Deberá completar una tabla de verdad que se corresponda con la función lógica del mismo.
Figura: 

Tabla 3-9: Ejercicio 8

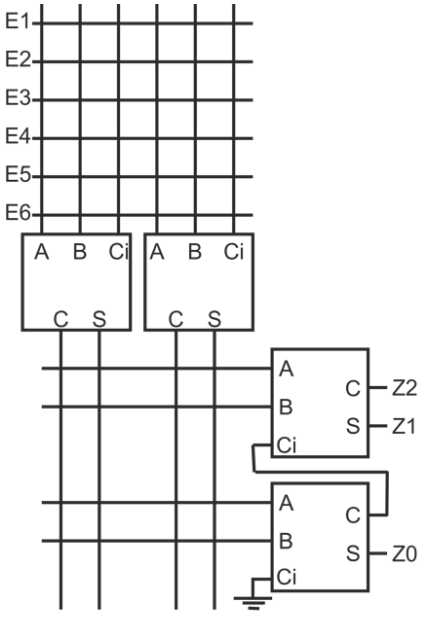
Ejercicio 9
Concepto: Suma
Descripción: El usuario encontrará una figura de un circuito de seis entradas. Deberá completar las conexiones para el correcto funcionamiento del mismo si se desea contar el número de entradas que están activas. El objetivo de este ejercicio es reforzar la comprensión del ejercicio anterior.
Figura: 

Tabla 3-10: Ejercicio 9

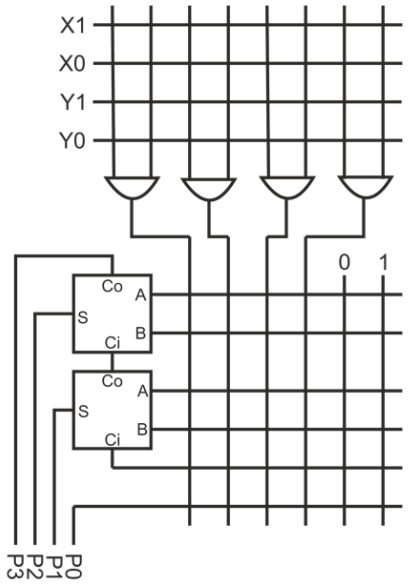
Ejercicio 10
Concepto: Multiplicación
Descripción: El usuario encontrará una figura de un circuito. Deberá completar las conexiones para el correcto funcionamiento del mismo como multiplicador de dos números de dos bits.
Figura: 

Tabla 3-11: Ejercicio 10

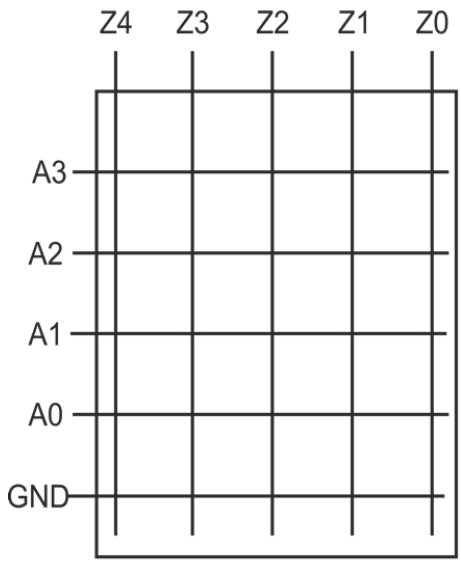
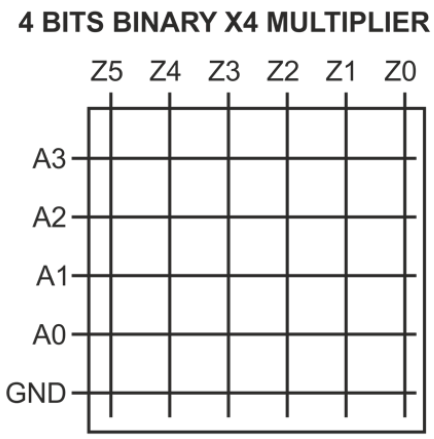
Ejercicio 11 y 12	
Concepto: Multiplicación	
Descripción: El usuario encontrará una figura de un circuito. Deberá completar las conexiones para el correcto funcionamiento del mismo como multiplicador de números binarios de cuatro bits por dos o por cuatro.	
Figura Ejercicio 11 	**Figura Ejercicio 11** 

Tabla 3-12: Ejercicios 11 y 12

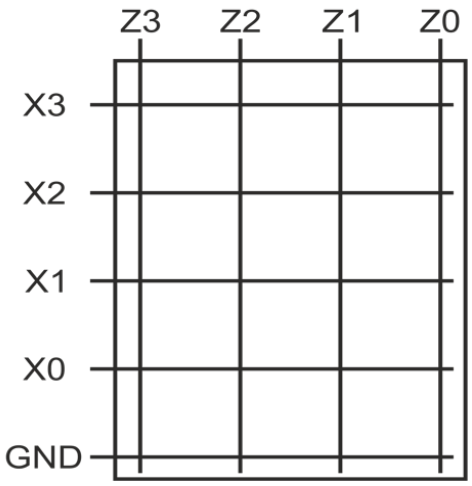
Ejercicios 13 y 14
Concepto: División
Descripción: El usuario encontrará una figura de un circuito. Deberá completar las conexiones para el correcto funcionamiento del mismo como divisor de números binarios de cuatro bits entre dos o cuatro.
Figura: 

Tabla 3-13: Ejercicios 13 y14

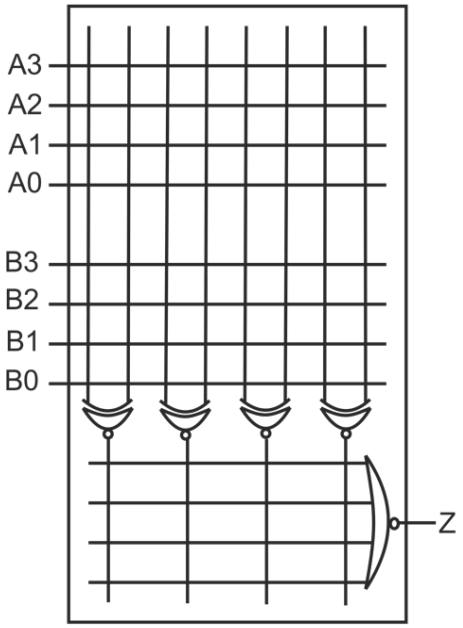
Ejercicio 15
Concepto: Comparación
Descripción: El usuario encontrará una figura de un circuito. Deberá completar las conexiones para el correcto funcionamiento del mismo para que Z=1 si A=B. El objetivo de este ejercicio es comprender el concepto de la comparación binaria.
Figura: 

Tabla 3-14: Ejercicio 15

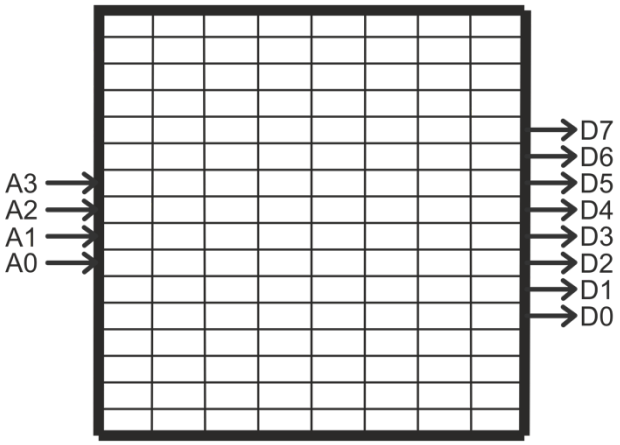
Ejercicios 16, 17, 18
Concepto: Tablas de *Look-Up*
Descripción: El usuario deberá rellenar el contenido de una tabla de Look-Up según las condiciones que se le indique en el enunciado. El objetivo de este ejercicio es comprender como se pueden utilizar las memorias ROM para realizar funciones lógicas.
Figura: 

Tabla 3-15: Ejercicios 16, 17, 18

4 Desarrollo

En este apartado abordaremos todos los conceptos relacionados con el desarrollo de la aplicación móvil:

- Pasos previos al desarrollo.
- Herramientas utilizadas.
- Elementos principales de una aplicación.
- Desarrollo de las diferentes funcionalidades de la aplicación.
- Clases Android utilizadas.

4.1 Pasos previos

Antes de comenzar con el desarrollo de la aplicación móvil se han llevado a cabo una serie de acciones con el fin de facilitar el proceso:

- Lectura de libros sobre programación Android.
- Desarrollo de pequeñas aplicaciones que después sirviesen para el desarrollo de este proyecto.
- Realización de pequeños tutoriales para familiarizarnos con el entorno de programación.

4.2 Herramientas de trabajo

Las herramientas utilizadas para el desarrollo del proyecto son las siguientes:

- Software de desarrollo de aplicaciones para Android (Android Studio).
- Ordenador personal.
- Smartphone Android.
- Emulador incluido en Android Studio.
- Software para la edición de imágenes.

4.3 Elementos principales de una aplicación Android

4.3.1 Actividad

Una actividad es un componente de la pantalla con el que los usuarios pueden interactuar para realizar una acción. A cada actividad se le asigna una ventana que representa la interfaz de usuario, y que normalmente ocupa todo el espacio disponible.

Una aplicación normalmente está formada por un conjunto de actividades, vinculadas entre sí de manera flexible. Cada una puede iniciar a su vez nuevas actividades.

Cada actividad está formada por dos partes. Una parte lógica que correspondería al archivo Java y una parte gráfica que correspondería al archivo xml. La parte lógica es aquella en la que se coloca y manipula el código desarrollado y la parte gráfica es aquella en la que se disponen los elementos que conforman la actividad y que serán manipulados según el código de la parte lógica.

Ambas partes quedan enlazadas a través del comando "*SetContentView*" a través del cual ligamos un archivo Java con su correspondiente archivo xml.

Las actividades se inician con el método "*onCreate*". Cada vez que se inicia una actividad nueva la anterior se detiene, pero se conserva en una pila. Cuando el usuario deja de interactuar con la actividad actual y pulsa el botón "*Atrás*" la destruye.

Toda actividad Android cuenta con un ciclo de vida que es importante comprender para realizar una correcta gestión de los procesos que está llevando a cabo nuestra aplicación.

Debemos considerar que este ciclo de vida tiene tres estados:

- **Reanudada**: la actividad se encuentra en el primer plano de pantalla y tiene la atención de usuario.
- **Pausada**: otra actividad se encuentra en primer plano y tiene la atención del usuario, pero esta todavía está visible. Esta actividad puede que se encuentre parcialmente visible en la pantalla o que no la cubra completamente. Una actividad pausada está completamente viva pero el sistema puede suprimirla en algún momento si fuese necesario.
- **Detenida**: la actividad ha quedado totalmente oculta debido a otra actividad.

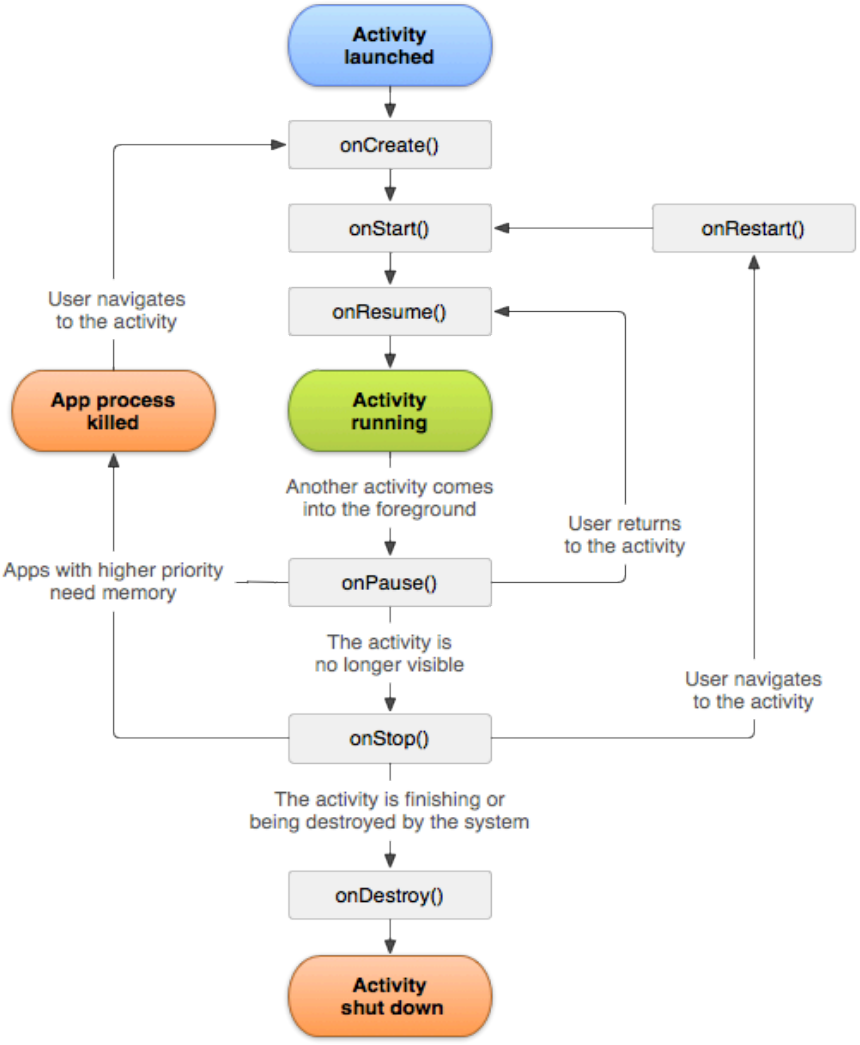


Figura 4-1: Ciclo de vida de una actividad en Android (según Android Developers)

4.3.2 Vistas

Una vista es un objeto que pertenece a la clase "*android.view.View*". Este objeto es una estructura de datos cuyas propiedades contienen los datos de la capa, la información específica del área rectangular de la pantalla, y que nos permite establecer el layout.

Este tipo de clase es muy útil ya que es la base para los widgets, que son unas subclases ya implementadas que sirven para dibujar elementos en la pantalla previamente definidos como cuadros de texto, botones, texto plano, etc.

4.3.3 Manifiesto de Android

Todas las aplicaciones deben tener un archivo con el nombre exacto AndroidManifest.xml en su directorio raíz. Este archivo proporciona información esencial sobre la aplicación junto a la información necesaria para la ejecución de la misma.

También nombra el paquete Java de la aplicación, que sirve como un identificador único.

Entre la información que contiene encontramos lo siguiente:

- Describe los componentes de la aplicación, como las actividades, los servicios, los receptores de mensajes y los proveedores de contenido que la integran.
- Determina los procesos que alojan a los componentes de la aplicación.
- Determina los permisos que debe tener la aplicación.
- Declara el nivel mínimo de Android API que requiere la aplicación.
- Enumera las bibliotecas con las que debe estar vinculada la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />

    </application>

</manifest>
```

Figura 4-2: Ejemplo AndroidManifest.xml

4.4 Esquema de la aplicación

A continuación se muestra un esquema de la estructura que tiene la aplicación que se ha desarrollado:

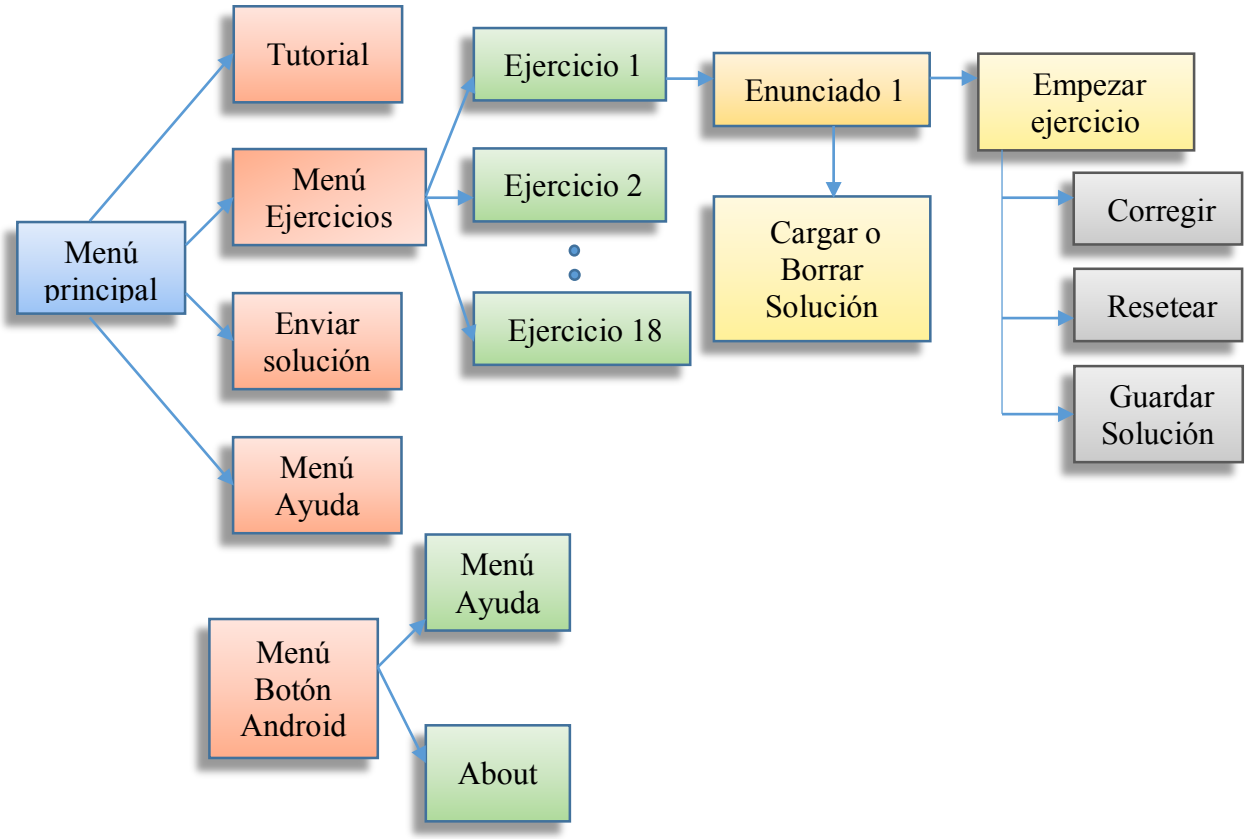


Figura 4-3: Esquema de la aplicación

4.5 Menús de la aplicación

Esta aplicación se dispone en torno a cuatro menús principales. Estos menús están desarrollados de una manera simple a través de los botones predeterminados de Android.

Los menús que se han desarrollado son los siguientes:

- **Menú principal**: es el menú que encontramos en el momento en el que abrimos la aplicación. Está compuesto por cuatro botones a través de los cuales podemos acceder al tutorial de la asignatura, al menú de ejercicios, a la opción de envío de soluciones o al menú de ayuda.
- **Menú de ejercicios**: es un menú accesible desde el menú principal con una apariencia similar a este, pero con la diferencia de que en él podemos hacer "*Scroll*" para poder ver todos los botones disponibles. A través de ellos podemos acceder a los enunciados de los ejercicios.
- **Menú botón Android**: este menú es accesible desde cualquier parte de la aplicación, menos desde los ejercicios de formatos de números. Se despliega en cualquier pantalla dándonos acceso a la sección "*Acerca de*" y al menú de ayuda.
- **Menú de ayuda**: este menú es accesible desde el menú principal, o desde el menú desplegable por el botón Android. En él se muestra una lista de elementos que contienen los diferentes consejos que pueda necesitar el usuario para usar adecuadamente la aplicación.

A continuación se muestran imágenes de los distintos menús mencionados previamente:

Figura 4-4: Menú botón Android

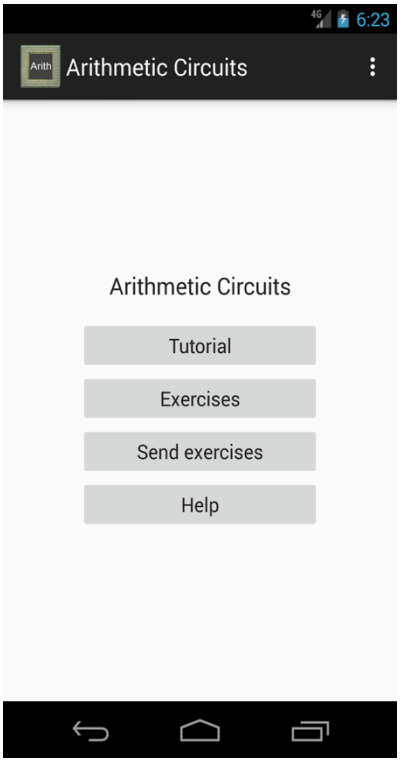


Figura 4-5: Menú principal

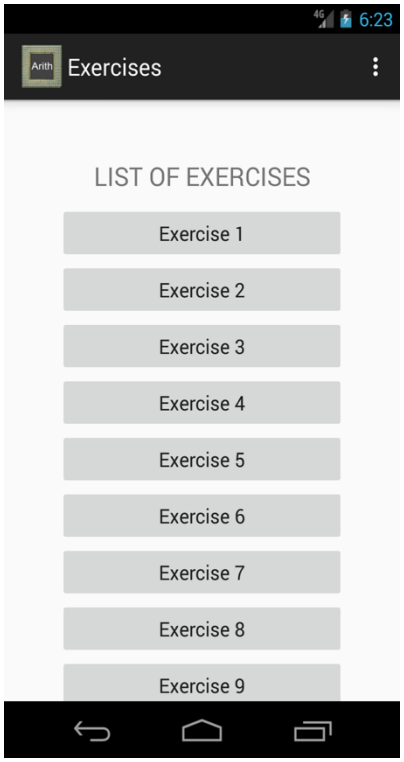


Figura 4-6: Menú ejercicios

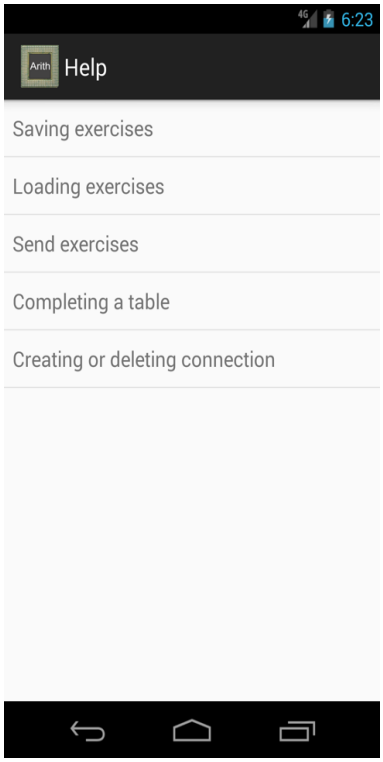


Figura 4-7: Menú ayuda

4.6 About

Esta es la primera actividad de la aplicación que se ha desarrollado debido a su simplicidad.

Consta simplemente de una imagen mostrada a través de un "*ImageView*". El usuario no interactúa en ningún momento con esta actividad.

La imagen que se muestra contiene la información relevante sobre el desarrollador, lo que normalmente se conoce como "*Acerca de*".

El diseño de esta actividad se ha realizando siguiendo el modelo mostrado en otras aplicaciones desarrolladas por DSLab.

A continuación se muestra el resultado que se puede observar en esta sección:

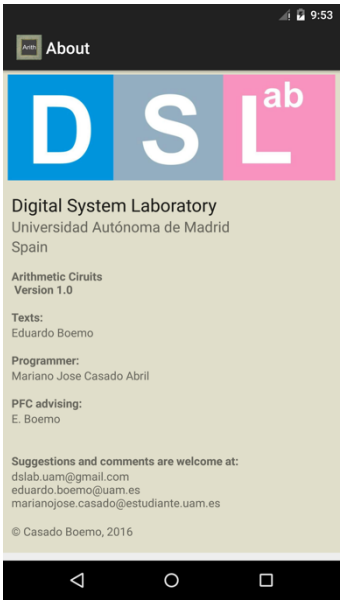


Figura 4-8: Apariencia About (*Acerca de*)

4.7 Tutorial

Como se menciona en la sección de diseño de este proyecto, la aplicación cuenta con una sección de tutorial en forma de diapositivas donde se presenta el material relevante de la asignatura relacionado con los Circuitos Aritméticos.

El usuario puede acceder a este tutorial desde el menú principal. Una vez que el usuario accede puede observar en la pantalla un cuadro de diálogo en el cual se le informa de que, para cambiar de diapositiva, debe desplazar la imagen a izquierda o derecha según quiera avanzar o retroceder.

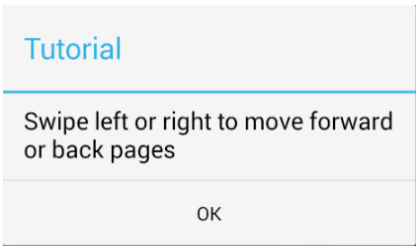


Figura 4-9: Información tutorial

Esta actividad se ha realizado sobre una única vista en la cual se mostrarán las imágenes. Estas imágenes se cargan a través de un "*ImageAdapter*" cada una con un identificador. Ese "*ImageAdapter*" será el utilizado para visualizar las imágenes.

El usuario avanzará o retrocederá haciendo que aumente o disminuya un contador utilizado para saber en que imagen nos encontramos. En el momento en el que el usuario se encuentre en la primera imagen solo podrá avanzar, e igualmente, si se encuentra en la última solo podrá retroceder.

4.8 Desarrollo ejercicios tipo tabla

Para desarrollar cada ejercicio de este tipo es necesario la implementación de dos componentes distintos, la vista y la actividad.

4.8.1 Vista

En este tipo de ejercicios la vista se desarrolla en un archivo xml.

En primer lugar hemos tenido que definir un *layout* de tipo lineal que nos sirve para que los elementos se distribuyan uno debajo de otro. En su interior debemos definir otro, de tipo tabla, que contiene 10 filas en el caso del ejercicio 8, y 18 en el resto de ejercicios de este tipo.

Cada fila contiene los elementos de texto utilizados para darle la apariencia deseada a las tablas. En el caso del ejercicio 8 son los diferentes valores que toman las variables, y en el caso del resto de ejercicios son los números de las posiciones que hay en la memoria.

Todas las filas contienen a su vez botones del tipo "Toggle Button", excepto la última, que se ha utilizado para distribuir los botones de tipo "Button" que serán utilizados por el usuario para acceder a la corrección, reseteo, solución o guardado de los ejercicios.

A continuación se muestra imágenes de la apariencia de este tipo de ejercicios:

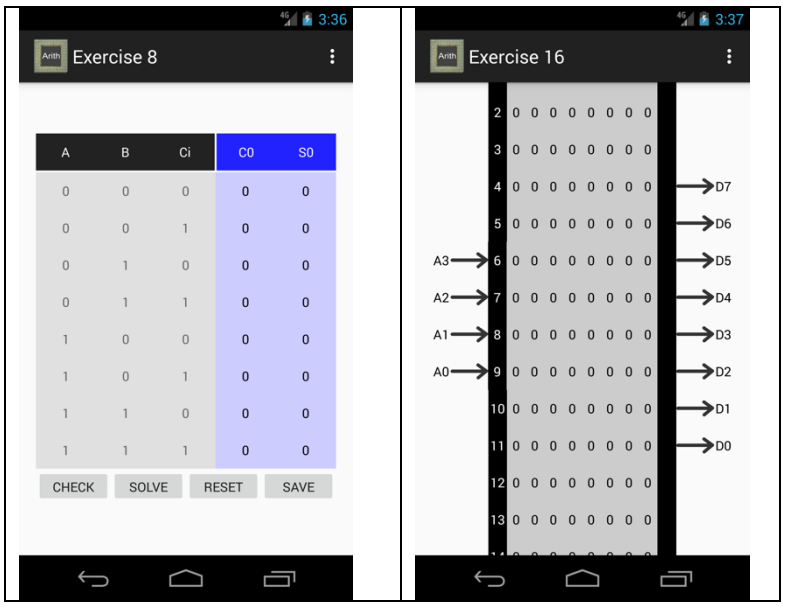


Figura 4-10: Apariencia Ejercicios de Tipo Tabla

4.8.2 Actividad

En esta parte de los ejercicios debemos ir gestionando los datos introducidos por el usuario. Para ello cada "*ToggleButton*" de la vista tiene un identificador único que será utilizado para saber que botón está modificando el usuario.

Se han definido dos vectores principales:

- ToggleButton: vector del tipo "*ToggleButton*" que se utiliza para modificar los valores de la vista.

- Vector OUT_TB de tipo "*boolean*" en el que se copiarán los datos del primer vector para que cuando los modifiquemos no cambie la vista.

El problema del primer vector es que en el momento en el que modifiquemos su valor en el código, este también cambiará en la vista. Por lo tanto gracias al segundo vector, podemos leer los datos introducidos por el usuario, sin cometer el error de modificarlos hasta que sea necesario que el usuario perciba los cambios.

Para copiar la información del primer vector en el segundo se ha implementado el método "*Guardar_TB*". En él copiamos el estado de los "*ToggleButon*" en el vector de tipo "*boolean*", utilizando la función "*isChecked*". Se ha aprovechado para implementar el registro de las preferencias compartidas en su interior, lo cual se explicará más adelante.

Una vez que tenemos ya la manera de gestionar los datos introducidos por el usuario se pueden implementar las diferentes funcionalidades a las que se podrá acceder:

- **Corrección**: se realizará la comprobación de los datos introducidos por el usuario. Se explicará en una sección posterior.
- **Reseteo**: utilizando la función "*setChecked*" con valor false, conseguimos poner todos los *ToggleButton*" a "0".
- **Solución**: utilizamos la misma función que en el reseteo, pero en lugar de utilizar el valor false, introducimos unos valores previamente definidos que forman parte de la solución.
- **Guardado**: se llevará a cabo el almacenamiento de los datos introducidos en el ejercicio. Se explicará en una sección posterior.

4.9 Desarrollo ejercicios de formatos de números

Para desarrollar cada ejercicio de este tipo es necesario la implementación de dos componentes distintos, la vista y la actividad.

4.9.1 Vista

En este tipo de ejercicios la vista se desarrolla en un archivo xml.

En primer lugar hemos tenido que definir un *layout* de tipo lineal que nos sirve para que los elementos se distribuyan uno debajo de otro. En su interior debemos definir otro de tipo tabla que contiene tantas filas como datos tenga el ejercicio.

Cada fila contiene un texto plano en el cual se muestra el dato del enunciado junto a un componente "*EditText*" en el cual el usuario puede introducir los datos requeridos.

La última fila se ha utilizado para distribuir los botones de tipo "*Button*" que serán utilizados para acceder a la corrección, reseteo, solución o guardado de los ejercicios.

Dentro de la vista se ha podido fijar el número máximo de caracteres que el usuario puede introducir en cada cuadro de texto, lo cual es muy útil a la hora de realizar la corrección.

4.9.2 Actividad

Dentro de esta parte de los ejercicios debemos ir gestionando los datos introducidos por el usuario. Por ello cada cuadro de texto "*EditText*" de la vista cuenta con un identificador único, para saber cual está modificando el usuario.

Los datos introducidos en estos cuadros de texto se almacenarán en un vector de tipo "*String*", que tiene un tamaño igual al número de cuadros de texto.

Uno de los problemas a los que hay que enfrentarse en este tipo de ejercicios es como guardar los datos del ejercicio sobre los cuadros de texto si el usuario sale del ejercicio sin guardar. Para ello se han utilizado el método "*addTextChangedListener*" a través del cual podemos hacer que cuando el usuario deje de escribir en uno de los cuadros de texto, el dato que ha introducido se escriba automáticamente en el archivo de preferencias. La función utilizada para guardar las preferencias se explicará en un apartado posterior.

El método anterior también es útil ya que cuando el usuario deja de escribir, los datos que ha escrito se almacenan automáticamente en el vector de tipo "*String*" para su posterior manipulación.

Una vez que tenemos ya la manera de gestionar los datos introducidos por el usuario se pueden implementar las diferentes funcionalidades a las que se podrá acceder:

- **Corrección**: se realizará la comprobación de los datos introducidos por el usuario. Se explicará en una sección posterior.
- **Reseteo**: utilizando la función "*setText*" con valor null, conseguimos poner todos los "*EditText*" en su valor por defecto, en este caso sin texto.
- **Solución**: utilizamos la misma función que en el reseteo, pero en lugar de utilizar el valor null, introducimos unos valores previamente definidos que forman parte de la solución.
- **Guardado**: se llevará a cabo el almacenamiento de los datos introducidos en el ejercicio. Se explicará en una sección posterior.

4.9.3 Diseño teclados personalizados

Una vez que se habían creado este tipo de ejercicios, se pudo comprobar como resultaba sencillo para el usuario introducir un dato, siempre que éste fuese un dato decimal.

El problema surgía en el momento en el que el usuario debía teclear datos binarios o hexadecimales, debido a que resulta tedioso utilizar el teclado convencional del teléfono.

Por ello se decidió diseñar dos teclados, uno binario y otro hexadecimal, que contuviesen simplemente las teclas que el usuario necesitaba, sin caracteres adicionales.

Para ello hay que crear dos tipos de archivos:

Archivo xml

En este archivo se definen una serie de filas, con los botones que va a contener cada una. Cada botón va asociado a unas etiquetas que indican el carácter que se está introduciendo y el símbolo de cada tecla, que puede ser texto o una imagen.

Clase CustomKeyboard

En este archivo debemos definir que ocurrirá cuando el usuario pulse cada una de las teclas.

Es una operación sencilla ya que simplemente debemos usar el código asignado a cada tecla y asociarlo a una acción concreta que se aplicará a un cuadro de texto en el caso de las teclas especiales como borrado, avance, retroceso, etc. En el caso de las teclas numéricas o de texto simplemente debemos introducir el carácter correspondiente al código de la tecla. Estas acciones se realizan a través del método "*onKey*".

Con esta clase también conseguimos decirle al teclado si debe estar visible o no.

Para que el uso de este tipo de teclados sea efectivo debemos declarar nuestro teclado personalizado en cada archivo Java de los ejercicios.

Se han utilizado los métodos "*hideCustomKeyboard*" y "*showCustomKeyboard*" para gestionar la visibilidad del teclado desde la actividad de ejercicios haciendo que este aparezca en el momento en el que el usuario pulse sobre el "*EditText*" y desaparezca cuando el usuario pulse el botón "Back".

Debemos utilizar el método "*registerEditText*" definido en esta clase en cada archivo Java de ejercicios para asignar que cuadros de texto queremos que funcionen con un teclado personalizado.

Una vez realizado este proceso nuestros teclados personalizados estarían disponibles.

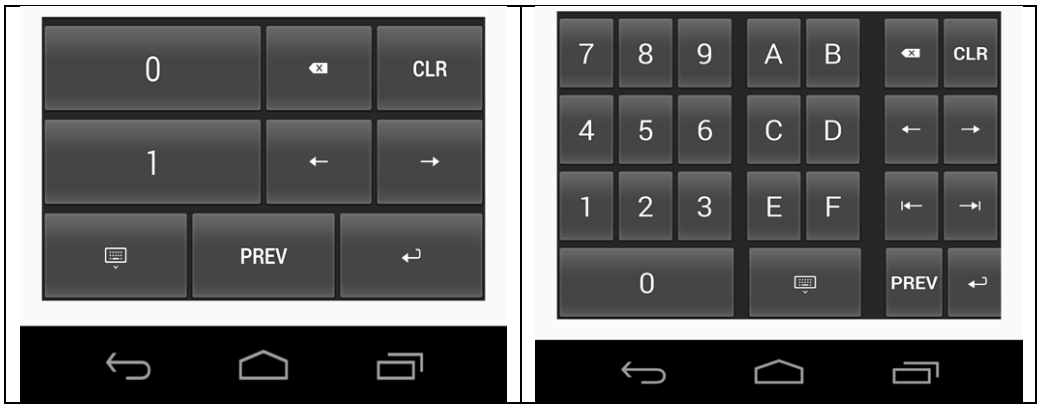


Figura 4-11: Teclados personalizados binario y hexadecimal

4.10 Desarrollo ejercicios de conexiones

Este tipo de ejercicios solo cuenta con un archivo de tipo Java donde se desarrolla tanto el código funcional como su apariencia.

Para abordar el diseño de estos ejercicios ha sido importante tener en cuenta que los circuitos debían tener unas proporciones similares a las de una pantalla estándar de teléfono móvil. Para ello se han diseñado creando un documento en el software de diseño gráfico con unas dimensiones similares a las de la pantalla del dispositivo donde se han realizado las pruebas.

El mayor problema que ha surgido a la hora de diseñar los circuitos ha sido ajustarlos para que estos se dispongan de una manera vertical, ya que si su tamaño no era el adecuado para la pantalla, se podría desplazar la vista verticalmente utilizando el "*Scroll*" disponible.

Las conexiones han sido diseñadas de tal manera que la distancia entre dos consecutivas sea conocida ya que, gracias a la herramienta de diseño, podemos ayudarnos de una serie de puntos distribuidos en forma de malla separados a una distancia definida por nosotros. De este modo y según las exigencias de cada ejercicio, las conexiones estarán a 1, 0.75 o 0.5 cm de distancia.

Conociendo la distancia entre conexiones en cm, el ancho de la imagen en cm y el ancho de la pantalla en píxeles, si asumimos que cada imagen ocupará aproximadamente el 90% de la pantalla, podemos obtener la distancia en pixeles entre conexiones:

$$Relacion\ \ cm\ y\ píxeles = \frac{90\%\ del\ ancho\ de\ pantalla\ en\ píxeles}{Ancho\ de\ imagen\ en\ cm}$$

$$Distancia\ conexiones\ en\ píxeles = Relación\ cm\ y\ píxeles * Distancia\ conexiones\ en\ cm$$

Para obtener el punto en el que se localiza la primera conexión debemos calcular donde se localiza la esquina superior izquierda de la imagen. Para ello cargaríamos la imagen en un mapa de bits y la localizaríamos en la posición donde se sitúa el inicio de los ejes X e Y de la pantalla para que esta quede centrada. Lo lograríamos dividiendo la diferencia entre el ancho y alto de la pantalla con el ancho y alto de la imagen respectivamente entre dos.

Conociendo los datos anteriores ya podríamos localizar el primer punto de las conexiones situado en la esquina superior izquierda. El resto de puntos se localizarán gracias a un bucle en el que avanzaremos horizontal o verticalmente según las necesidades del ejercicio, definiendo la coordenada X e Y de cada conexión.

Una vez localizadas las conexiones podemos dibujarlas. Para ello se ha utilizado el método "*onTouchEvent*". Gracias a este método y a un objeto del tipo "*MotionEvent*" podemos conocer donde ha pulsado el usuario.

Como no se puede esperar que el usuario pulse exactamente en el punto de interconexión de dos líneas del circuito se ha habilitado una zona activa alrededor de cada punto de manera que en el momento en el que el usuario pulse dentro de esta zona activa, se habilitará o deshabilitará la conexión. Cada conexión activa se marcará con un punto azul con un tamaño definido previamente.

Para dibujar o eliminar el punto hay que recurrir al método "*onDraw*". Este método se invoca desde el método "*onTouchEvent*". Cada vez que se va a dibujar o eliminar un punto se actualiza el estado de la conexión, se dibuja el punto concreto y se actualiza el archivo de preferencias para conseguir que si el usuario abandona el ejercicio sin guardar la solución los datos permanezcan intactos cuando reanude la resolución del ejercicio. El método de preferencias se explicará posteriormente.

A la hora de disponer unos botones para que el usuario accediese a las funcionalidades disponibles en la resolución como guardar, resetear o corregir, nos encontramos con el

problema de que este tipo de ejercicios no cuenta con un archivo xml donde se definan los botones.

Para solucionarlo se ha optado por dibujar unos botones con unas zonas activas similares a las de las conexiones, de manera que cuando el usuario pulse en la zona activa, se ejecuten los métodos de guardado, corrección o reseteo similares a los de los otros ejercicios.

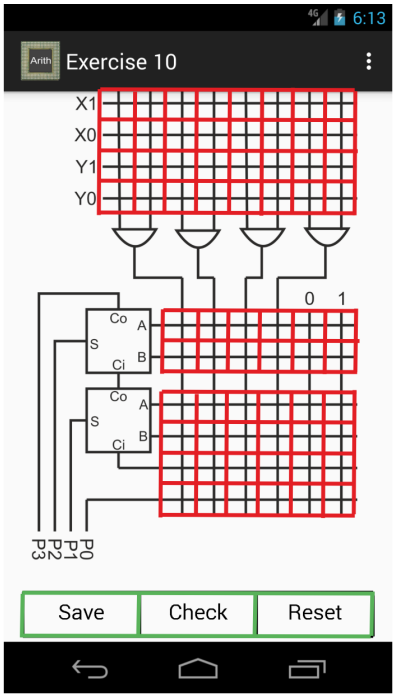


Figura 4-12: Localización de las zonas activas (en rojo las de las conexiones, en verde las de los botones)

4.11 Adaptación a diferentes tamaños de pantalla

Al diseñar una aplicación para dispositivos móviles debemos conseguir que esta se visualice correctamente en cualquier tipo de pantalla, es decir, conseguir adaptarla a diferentes densidades y tamaños.

Para ello se han diseñado las vistas de dos maneras diferentes según si los ejercicios son de formatos de números o de tipo tabla, o ejercicios de dibujar conexiones.

4.11.1 Adaptación de pantalla en ejercicios de formatos de números y de tipo tabla

Para adaptar las vistas de este tipo de ejercicios Android nos da la posibilidad de crear una serie de carpetas, cada una de las cuales hace referencia a un tamaño y a una densidad de pantalla diferente.

Podemos desarrollar una vista diferente de cada ejercicio según el tamaño y densidad de pantalla que deseemos.

Por ello, para solucionar el problema de los diferentes tamaños, se han creado dos archivos xml distintos para cada ejercicio, uno para ser mostrado en pantallas normales y otro para ser mostrado en pantallas de tipo "*large*".

Ambos archivos tienen una estructura similar. Lo que los diferencia es que en ellos se han aplicado unos tamaños de fuente y de márgenes diferentes, según si la pantalla en la que se van a mostrar es mayor o menor.

A su vez Android Studio nos permite utilizar varias versiones de las imágenes con diferentes densidades de pantalla, que serán utilizadas por los dispositivos según el valor de densidad requerido.

Para comprobar que nuestra aplicación se adapta perfectamente a cualquier tipo de pantalla se ha utilizado el emulador de Android, probándola en dispositivos con diferentes tamaños y densidades.

4.11.2 Adaptación de pantalla en ejercicios de conexiones

En este tipo de ejercicios la adaptación de las vistas no se hace con la creación de distintos archivos xml, sino en la propia actividad.

Para ello debemos crear una clase dentro del ejercicio de tipo View. Dentro de la misma si utilizamos el método "*onDraw*" podremos dibujar lo que queramos en la pantalla ya que se nos creará un objeto similar a un lienzo utilizando la clase "*Canvas*".

Para dibujar sobre este lienzo debemos conocer en que lugar de la pantalla queremos situar los objetos dibujados, que pueden ser textos, líneas, puntos o imágenes.

Para conocer las coordenadas disponibles en la pantalla debemos conocer el tamaño de esta. Para ello podemos utilizar un objeto del tipo "*DisplayMetrics*" a través del cual el sistema nos proporciona su resolución, que no es otra cosa que el número de píxeles en dirección vertical y horizontal.

Con los datos anteriores ya podríamos calcular la relación alto/ancho. Este valor será mayor o igual a 1, y se acercará a 1 cuanto más cuadrada sea la pantalla. Esta relación es importante, ya que las imágenes que insertemos deben tener una relación alto/ancho menor a la de la pantalla, debido a que la interfaz de usuario estará formada por:

- Una versión escalada de la imagen con la misma relación alto/ancho.
- Barra de Android.
- Botones.

En este caso se han diseñado imágenes para los ejercicios con una relación alto/ancho entre 1.3 y 1.5, ya que estos valores son menores que los valores típicos que suelen tener las pantallas de la mayoría de dispositivos.

Una vez que tenemos claro como proceder, utilizando el método "*onDraw*" previamente mencionado, creamos una versión escalada de la imagen que mantenga la relación alto/ancho de la original a través de un mapa de bits escalado, junto a unas líneas que definen los botones que tendrán los ejercicios.

Para la correcta visualización de las imágenes se han añadido unos márgenes iguales al 5% de la pantalla.
El problema surge cuando observamos que si utilizamos pantallas que tienen una relación alto/ancho menor a la de las imágenes, la visualización será incorrecta.

Para solucionar este problema se ha optado por realizar unos cambios, por los cuales, si la relación alto/ancho es menor a ciertos valores, aumentarán los márgenes de la pantalla, de forma que la zona en la que podemos distribuir la imagen tendrá una mayor relación alto/ancho. Estos valores son los siguientes:

- Para pantallas con relación alto/ancho menor de 1.4 se aplican márgenes del 15%.
- Para pantallas con relación alto/ancho menor de 1.52 se aplican márgenes del 10%.

Por último se ha adaptado la vista al tipo "*Scroll*" de manera que si por algún motivo, los botones no se visualizasen correctamente porque la pantalla fuese menos alta de lo esperado, el usuario pueda arrastrar la vista adaptándola a sus necesidades.

4.12 Gestión de datos en la aplicación

4.12.1 Método de Preferencias Compartidas

Las preferencias son simplemente los datos que una aplicación guarda para mejorar la experiencia del usuario. Estos datos se podrían almacenar de múltiples maneras como por ejemplo utilizando algún servicio de SQL, pero en este caso todo es mucho más sencillo, ya que Android proporciona un método llamado Preferencias Compartidas.

A través de este método cada preferencia se almacenará en forma clave-valor, es decir cada una de ellas se compondrá por un identificador único y un valor. A diferencia de otros métodos, este no guarda los datos en un archivo binario, sino en un xml.

Este proceso de manejo de preferencias necesita la declaración de un objeto del tipo "*SharedPreferences*", e indicar el nombre de la "colección" donde guardaremos los datos.

Para escribir los datos en los archivos de preferencias simplemente debemos utilizar un editor que debemos declarar previamente.

En el caso en el que queramos leer las preferencias simplemente debemos utilizar un método de lectura como getInt(), o getString() ,que son los que utilizamos en nuestro caso.

El método de preferencias compartidas se usa en nuestra aplicación en dos tipos de actividades:

- **Enunciados de los ejercicios:** Se utiliza simplemente para escribir el número de ejercicio. Este número se utilizará para cargar o eliminar la solución que el usuario seleccione.
- **Ejercicios:** Se utiliza para escribir el número de ejercicio que estamos solucionando junto a los datos relativos a la solución que el usuario desee guardar. A si vez es útil para que, en el caso en el que el usuario cierre un ejercicio sin resetear la solución, en el momento en el que vuelva a iniciarlo, los datos introducidos permanezcan intactos.

4.12.2 Guardar Solución de Ejercicio

El usuario puede acceder a este tipo de actividad desde cualquier pantalla de resolución de ejercicios pulsando el botón "*Save*".

En esta actividad se han definido una serie de vectores de distinta clase, en los cuales almacenaremos los datos que vamos a guardar para cada ejercicio. Como se ha mencionado anteriormente, para la lectura de los datos se utiliza el método de preferencias compartidas. Una vez que se han leído los datos estos se escriben en un fichero de una manera diferente para que sean fácilmente manipulables.

Se ha diseñado la vista de esta actividad como un diálogo en el cual el usuario puede introducir texto. Este diálogo muestra una cabecera predeterminada en la que se indica el número del ejercicio que se desea guardar junto a la localización en la que se guardará el archivo. Se indica con EM que el archivo se guarda en memoria externa. Se indica con IM que el archivo se guarda en memoria interna. El usuario puede completar el nombre del archivo con los datos que desee, y después debe pulsar el botón "*Save*" para completar el proceso de guardado.

Los datos se guardan por defecto en la memoria externa del dispositivo en el caso de tenerla. Si no dispone de ella se guardan en la memoria interna en una zona habilitada para el almacenamiento de los datos de la aplicación.

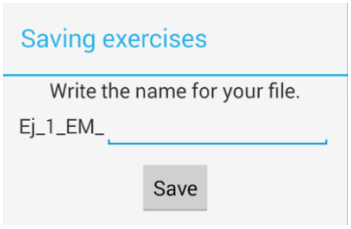


Figura 4-13: Guardar Ejercicios

4.12.3 Cargar o Eliminar Solución

Podemos acceder a esta funcionalidad desde cualquier pantalla en la que se muestre el enunciado de los ejercicios.

Cuando el usuario accede a esta opción se muestra una alerta en la cual debe seleccionar las soluciones que desee cargar o eliminar.

En el caso en el que desee eliminar las soluciones puede seleccionar múltiples archivos. Se muestra una alerta para confirmar la eliminación. En caso afirmativo, el archivo que contenía los datos de la solución desaparece completamente y no se puede volver a cargar.

En el caso en el que el usuario seleccione la opción de cargar una solución se muestra un mensaje de error si se ha seleccionado más de un archivo para cargar.

El proceso de carga se lleva a cabo mediante una actividad en la cual se han creado una serie de vectores similares a los creados en la actividad de guardado, donde se almacenan los datos que se van a cargar. Los datos son leídos del archivo guardado y almacenados en el vector correspondiente. Utilizando el método de gestión de datos anteriormente mencionado, se actualizan las preferencias con los valores guardados en esos vectores y se lanza la actividad

del ejercicio, de manera que, una vez iniciado, se carguen las nuevas preferencias que contienen los datos que previamente guardados.

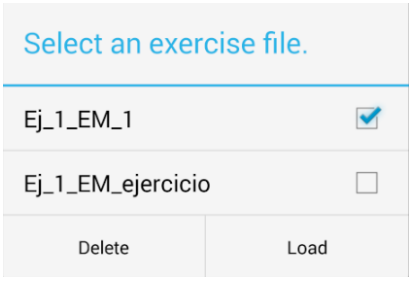


Figura 4-14: Cargar o Borrar Ejercicios

4.13 Envío de Soluciones

El usuario puede acceder a esta opción desde el menú principal de la aplicación. Esta funcionalidad se ha desarrollado directamente en el script Java correspondiente al menú principal.

Una vez que el usuario accede a esta opción se le muestra un diálogo de alerta en el que puede seleccionar las soluciones que desea enviar, siempre que previamente haya guardado al menos una. Una vez que las seleccione y pulse el botón de enviar se abre una ventana para seleccionar la aplicación de correo electrónico que desea utilizar para el envío.

En el momento en el que se ha seleccionado la aplicación de envío se adjuntan los archivos, se rellenan automáticamente tanto el asunto, como el cuerpo del correo que indica las instrucciones a seguir por el usuario receptor para poder cargar los archivos recibidos.

El usuario que reciba los archivos debe moverlos de la carpeta de descargas de su terminal a la carpeta creada por la aplicación. Esta carpeta solo está disponible en el momento en el que se haya guardado como mínimo una solución.

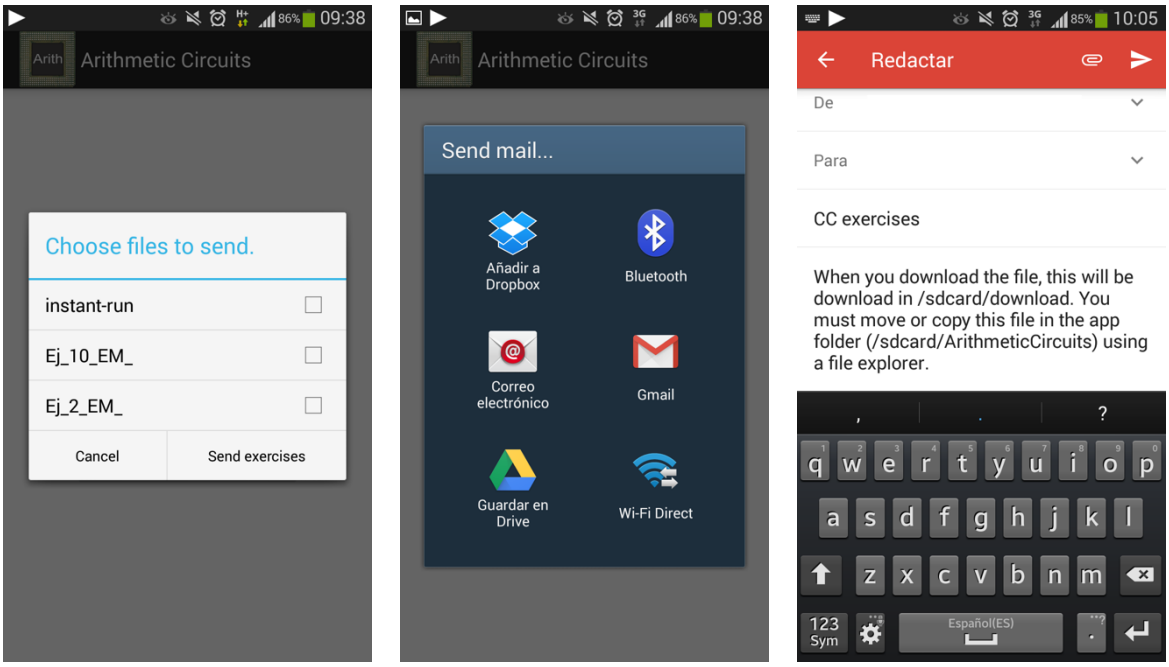


Figura 4-15: Envío de soluciones por correo electrónico

4.14 Comprobación de Soluciones

Las comprobación de soluciones se ha realizado de una manera diferente según el tipo de ejercicios que estemos comprobando.

4.14.1 Comprobación Ejercicios Formatos de Números

Al tratarse de ejercicios en los que el usuario tiene que introducir un valor numérico, solo hay una única solución correcta. Esto facilita mucho la comprobación de las soluciones. Para hacerlo aún más fácil se ha acotado el número de cifras que puede introducir el usuario de manera que si introduce un número de un tamaño menor al indicado se muestra un mensaje de error. Se ha hecho que los cuadros de texto con los que interactúa el usuario tengan un número máximo de caracteres a introducir.

En este caso la comprobación se realiza comparando cada elemento introducido en cada cuadro de texto con su valor correcto fijado en el código. En el caso incorrecto se indica en que respuesta está cometiendo el error el usuario.

4.14.2 Comprobación de Ejercicios de Tipo Tabla

Tanto en los ejercicios de la sección de tablas de Look-Up como en el ejercicio 8, la comprobación se realiza comparando los datos introducidos con una solución única.

Se realizan comprobaciones por fila utilizando las comparaciones de vectores que nos permite realizar Java. Comparamos cada fila con su valor correcto, añadiendo a una cadena de texto que filas no coinciden con la solución fijada, para que el usuario sepa en que parte del ejercicio ha cometido errores.

4.14.3 Comprobación de Ejercicios Conexiones

La comprobación de este tipo de ejercicios es la más complicada debido a que en algunos nos encontramos con que el usuario puede introducir múltiples soluciones que sean correctas. Esto ocurre en los ejercicios 9,10 y 15.

En el resto de ejercicios de conexiones, 11,12,13 y 14 la comprobación es mucho más sencilla ya que estos cuentan con una única solución válida, por lo tanto para comprobar que la solución introducida por el usuario es la correcta simplemente debemos definir un vector con los valores true o false adecuados según si la conexión debe estar activada o no, y compararlos con los valores de las conexiones pulsadas por el usuario.

En el caso de los ejercicios que no tienen solución única debemos abordar el problema de una forma distinta. Debemos agrupar las conexiones de una forma en que resulte más sencillo establecer las posibles combinaciones que pueda haber.

Para ilustrar este problema se pondrá como ejemplo la solución del ejercicio número 10 del que a continuación se muestran dos posibles versiones:

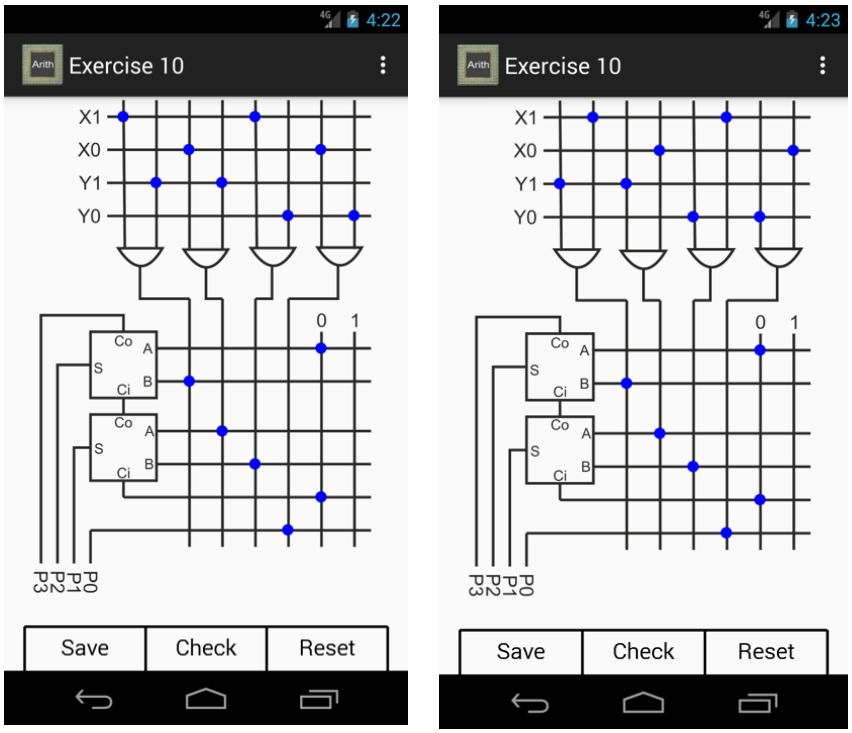


Figura 4-16: Dos posibles soluciones al ejercicio 10

Como podemos observar en este ejercicio hay dos partes diferenciadas, una superior y una inferior. La parte inferior tiene fácil solución ya que tiene una única solución válida.

El problema lo encontramos en la parte superior. A cada puerta AND hay que conectar dos entradas de las disponibles (X1, X0, Y1 o Y0). A su vez éstas se pueden conectar en cualquiera de las dos entradas de las AND, por ejemplo en la primera puerta podemos conectar X0-Y0, o Y0-X0. El problema se complica más ya que el usuario puede conectar cada par de entradas en la puerta que desee.

Para solucionar este problema en un principio se optó por distribuir las conexiones de la parte superior de forma vertical en lugar de hacerlo en horizontal como en la parte inferior. Con ello conseguimos aislar las conexiones de cada puerta AND consiguiendo así reducir el número de combinaciones posibles. Una vez que hemos reducido el problema, simplemente debemos comparar las pulsaciones introducidas por el usuario con unos vectores que contienen los valores de la respuesta correcta.

Debido a la complejidad de este tipo de corrección, finalmente, y siguiendo los consejos del profesor, se ha optado por eliminar el botón "*Check*" de los ejercicios con solución múltiple.

La solución correcta deberá ser contrastada con otros alumnos o con el profesor de la asignatura.

5 Integración, pruebas y resultados

5.1 Publicación en Google Play Store

Una vez que la aplicación está finalizada y hemos comprobado que no tiene errores, podemos proceder a su publicación en Google Play Store. Para ello primero debemos generar el fichero APK, y después podremos publicarla en la plataforma.

5.1.1 Generación fichero APK

Cuando tenemos la certeza de que la aplicación funciona correctamente debemos generar un archivo APK para que los usuarios puedan instalarla en sus dispositivos.

Hasta este momento la aplicación se estaba probando en modo "*Debug*" en un dispositivo móvil y en los diferentes dispositivos disponibles en el emulador proporcionado por Android Studio.

Para generar el fichero APK debemos utilizar el software Android Studio y acceder a la opción "*Build*". Una vez que accedemos debemos seleccionar la opción "*Generar APK firmada*".

Es importante resaltar que para que una aplicación sea admitida para su publicación en Google Play Store debe estar firmada con un certificado digital. El proceso de firma también es útil por motivos de seguridad, ya que así nos aseguraremos de que solo nosotros podemos modificar la aplicación.

Para firmarla debemos crear un *Key Store*. El proceso es sencillo ya que solo debemos introducir una contraseña y una serie de datos personales.

El archivo APK generado debería ser instalado por lo menos en un dispositivo móvil para comprobar el correcto funcionamiento de nuestra aplicación.

Tras realizar estos simples pasos ya dispondremos de un archivo APK para su publicación.

5.1.2 Cuenta de desarrollador

Para poder añadir una nueva aplicación a Google Play Store debemos tener acceso a una cuenta de desarrollador. En el caso de que no dispongamos de una podríamos obtenerla abonando 25 $. Tras el pago del dinero deberíamos rellenar una serie de datos relativos al desarrollador como por ejemplo el nombre, o un sitio web relacionado con la entidad o persona encargada de la aplicación.

En nuestro caso contamos con la cuenta cuenta perteneciente a DSLab cuyo nombre de desarrollador es "DSLab UAM".

5.1.3 Añadir la nueva aplicación a Google Play Store

Para poder publicar nuestra aplicación debemos acceder al sitio web "*Android Developer Console*" con nuestra cuenta de desarrollador.

Una vez que hemos accedido debemos seguir los siguientes pasos:

- Hacer clic sobre el botón "*Add New Application*" para abrir una ventana en la que debemos introducir el idioma, el título de la aplicación y deberemos adjuntar el archivo APK.
- Indicar si la versión de la aplicación es un test alfa, beta, o está en modo de producción.
- Acceder a la pestaña "*Store Listing*" en la cual deberemos indicar:
    - Descripción completa.
    - Texto de promoción.
    - Icono.
    - Pantallazos.
    - Categoría.
    - Privacidad.
    - Palabras clave, para facilitar la búsqueda de los usuarios.
- Acceder a la pestaña "*Pricing & Distribution*" para indicar el precio de la aplicación (en este caso es gratuita), y los países donde se distribuirá (se distribuye al máximo de países posible, 141 países y resto del mundo).

Una vez completado este proceso ya podemos publicar nuestra aplicación desactivando la opción de borrador.

5.1.4 Análisis de datos y estadísticas proporcionados por Google Play Store

Una vez publicada la aplicación y pasado un tiempo, podríamos acceder a una de las funcionalidades interesantes de Google Developer Console, que nos permite obtener datos acerca de el número de descargas efectuadas, el que países se han realizado, e incluso la frecuencia de las mismas.

El único problema es que esta aplicación se usará en el segundo cuatrimestre del curso académico 2016/2017, por lo tanto aún no tiene sentido acceder a estos datos.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Tras la realización del proyecto podemos concluir con que se han alcanzado los objetivos propuestos al inicio del mismo.

El objetivo principal de este proyecto era el diseño de una aplicación móvil para aprovechar las ventajas que nos ofrecen los dispositivos móviles actuales para fines docentes. El proyecto desarrollado permitirá a los alumnos ampliar sus conocimientos de una manera más simple y eficiente.

En cuanto a los objetivos de diseño se ha conseguido desarrollar:

- Un tutorial relativo a los circuitos aritméticos.
- 18 ejercicios interactivos sobre circuitos aritméticos.
- Función de corrección y guardado de las soluciones.
- Función para cargar o eliminar soluciones guardadas.
- Envío por correo electrónico de las soluciones guardadas.
- Un manual básico de ayuda para comprender el funcionamiento de la aplicación.
- Adaptación de las vistas para diferentes tipos de pantallas.

Desde el punto de vista de aprendizaje se ha conseguido obtener conocimientos suficientes sobre:

- Lenguaje xml.
- Lenguaje Java.
- Programación de aplicaciones Android.
- Manejo de Android Studio.
- Publicación de aplicaciones en Google Play Store.

6.2 Trabajo futuro

Gracias a la gran cantidad de ventajas que ofrece la tecnología móvil en el campo de la docencia, en el futuro se podría seguir mejorando esta aplicación con el fin de aumentar el número de usuarios.

Algunas de las posibles mejoras que se podrían realizar serían:

- Aumentar el número de ejercicios.
- Añadir más versiones de los ejercicios de memorias.
- Modernizar la interfaz de la aplicación.
- Añadir un sistema de comunicación entre usuarios.
- Añadir una versión para *SmartTV*.
- Añadir un posible test para comprobar los conocimientos obtenidos.
- Añadir un sistema para puntuar los ejercicios.
- Reducir el consumo de batería.
- Añadir soporte en otros idiomas ya que por ahora solo estará disponible en inglés.
- Añadir soporte para iOS.

Referencias

[1] Jesús Tomás Gironés, Marcombo Ediciones Técnicas, “El Gran Libro de Android”,2011.
[2] FAQsAndroid, “Curso de Programación en Android Para Principiantes”, Robert P.
[3] Maestros del Web, “Curso Android: Desarrollo de Aplicaciones Móviles”,2011.
[4] www.sgoliver.net, “Curso Programación Android”, Salvador Gómez Oliver,2011.
[5] “Introducción a Android”, Manuel Báez, Álvaro Borrego, Jorge Cordero, Luis Cruz, Miguel González, Francisco Hernández, David Palomero, José Rodríguez de Llera, Daniel Sanz, Mariam Saucedo, Pilar Torralbo, Álvaro Zapata .
[6] https://developer.android.com/
[7] http://stackoverflow.com
[8] http://es.androids.help/
[9] http://www.aprendeandroid.com
[10] http://androidos.readthedocs.io/en/latest/
[11] http://cursoandroidstudio.blogspot.com.es/
[12] http://www.javahispano.org/android/2012/6/4/crear-aplicaciones-compatibles-con-multiples-pantallas.html
[13] http://cincodias.com/cincodias/2015/02/01/lifestyle/1422792260_243066.html
[14] https://play.google.com/store

Glosario

API: Application Programming Interface

APK: Android Application Package

Bitmap: Mapa de Bits

CED: Circuitos Electrónicos Digitales

dpi: Dots Per Inch

DSLab: Digital System Laboratory

EPS: Escuela Politécnica Superior

hdpi: High Resolution Image

ldpi: Low Resolution Image

mdpi: Medium Resolution Image

UAM: Universidad Autónoma de Madrid

xhdpi: Extra High Resolution Image

Anexos

A Imágenes de los ejercicios incluidos

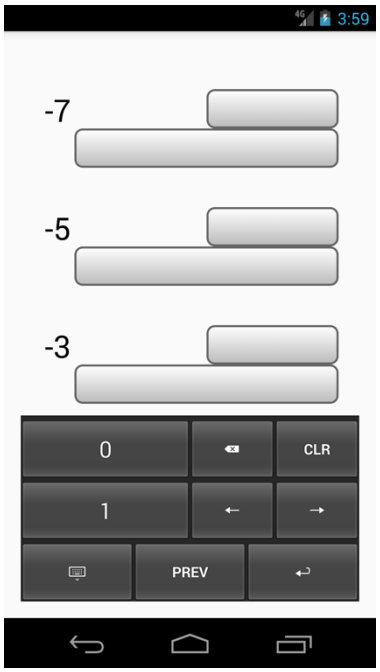


Figura 0-1: Ejercicio 1

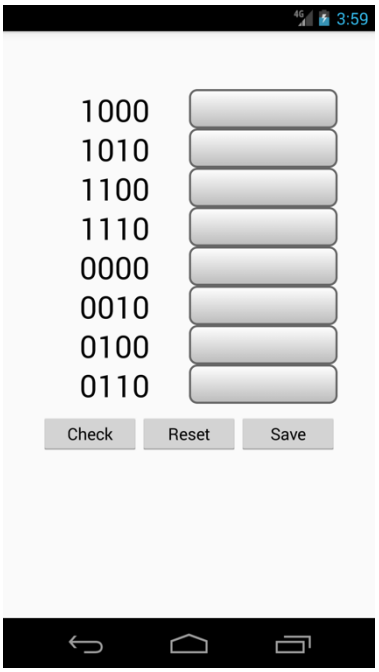


Figura 0-2: Ejercicio 2

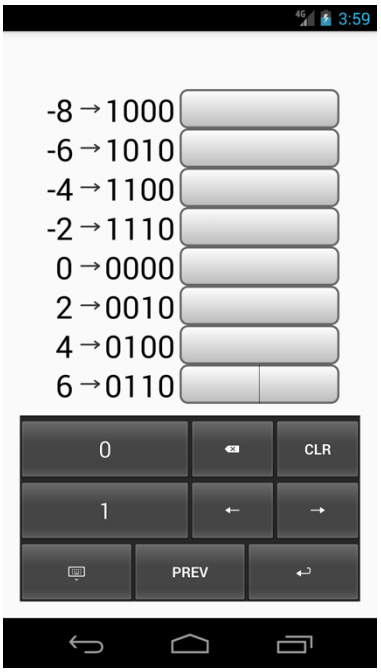


Figura 0-3: Ejercicio 3

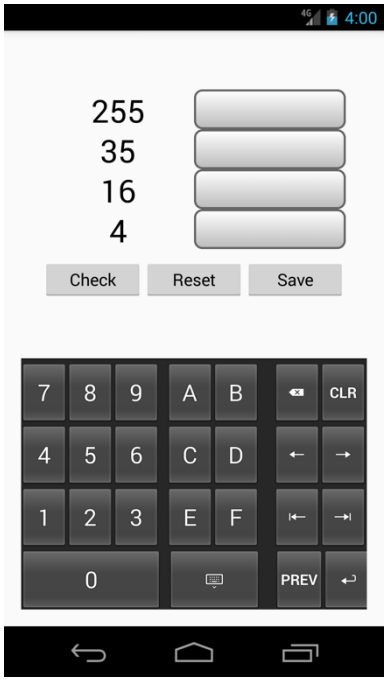


Figura 0-4: Ejercicio 4

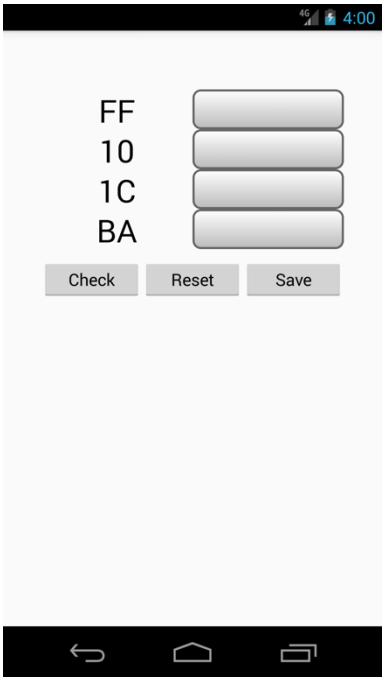


Figura 0-5: Ejercicio 5

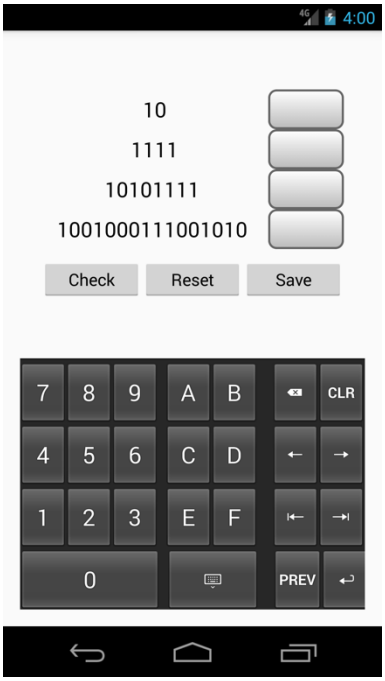


Figura 0-6: Ejercicio 6

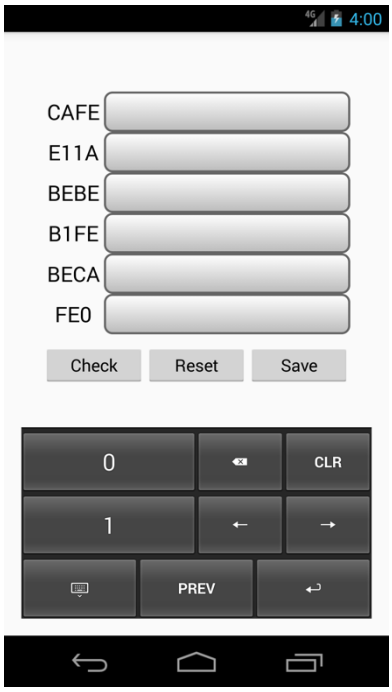


Figura 0-7: Ejercicio 7

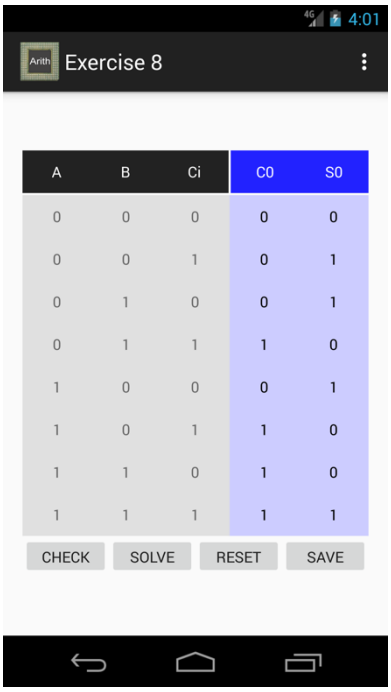


Figura 0-8: Ejercicio 8

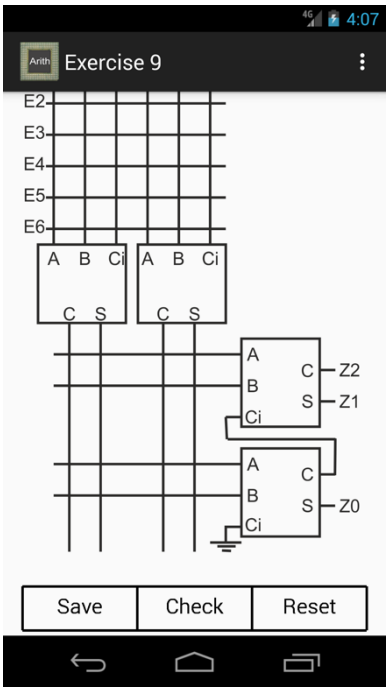


Figura 0-9: Ejercicio 9

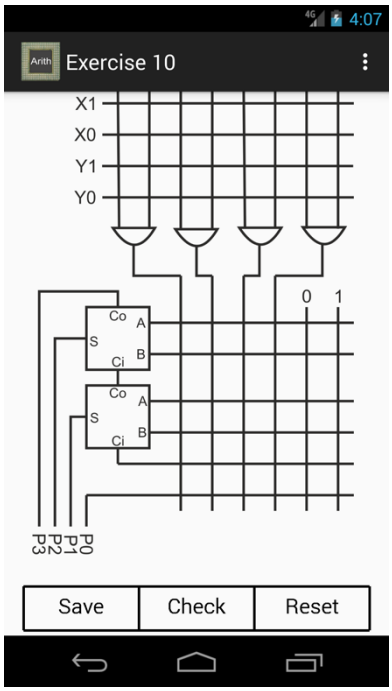


Figura 0-10: Ejercicio 10

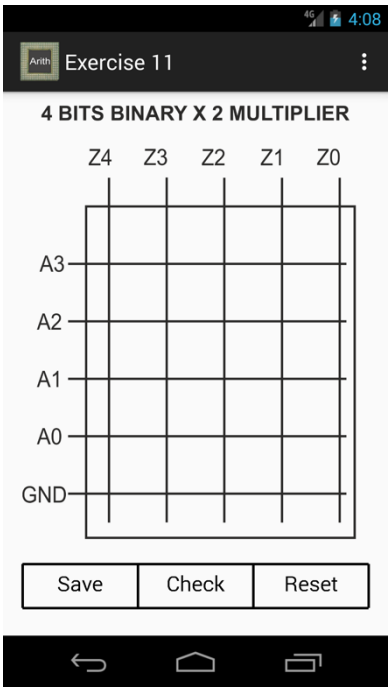


Figura 0-11: Ejercicio 11

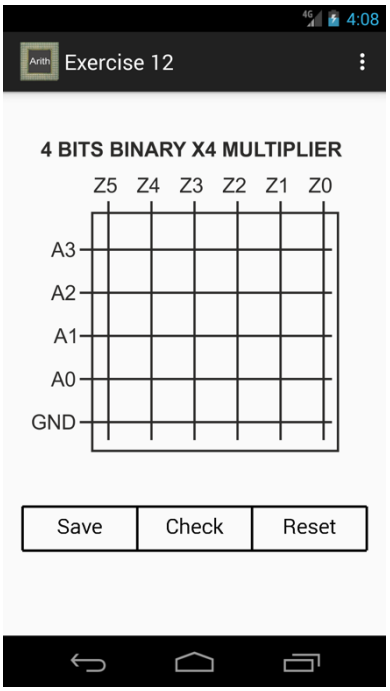


Figura 0-12: Ejercicio 12

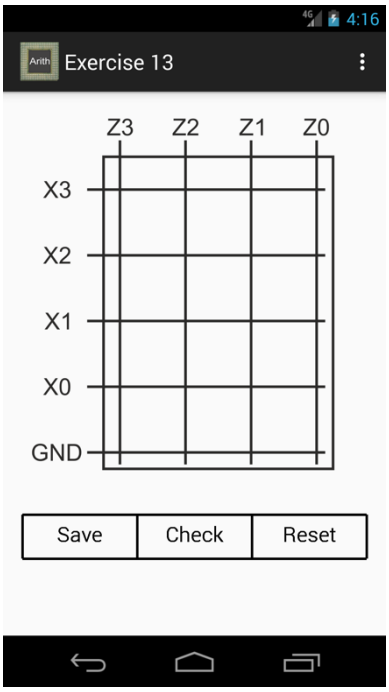


Figura 0-13: Ejercicios 13 y 14

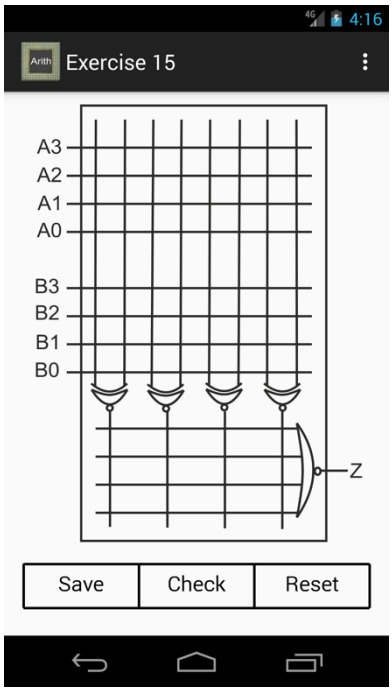


Figura 0-14: Ejercicio 15

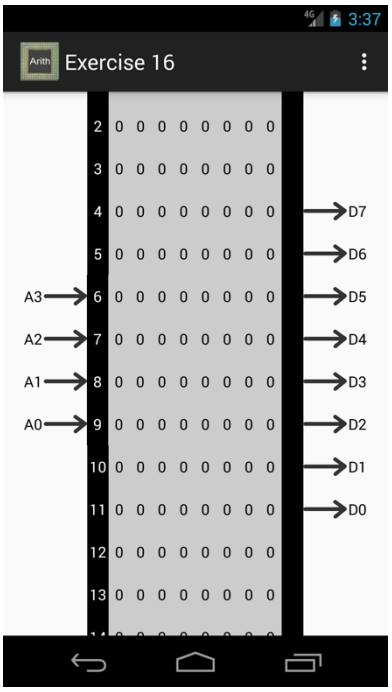


Figura 0-15: Ejercicios 16, 17 y 18

B Clases Java utilizadas

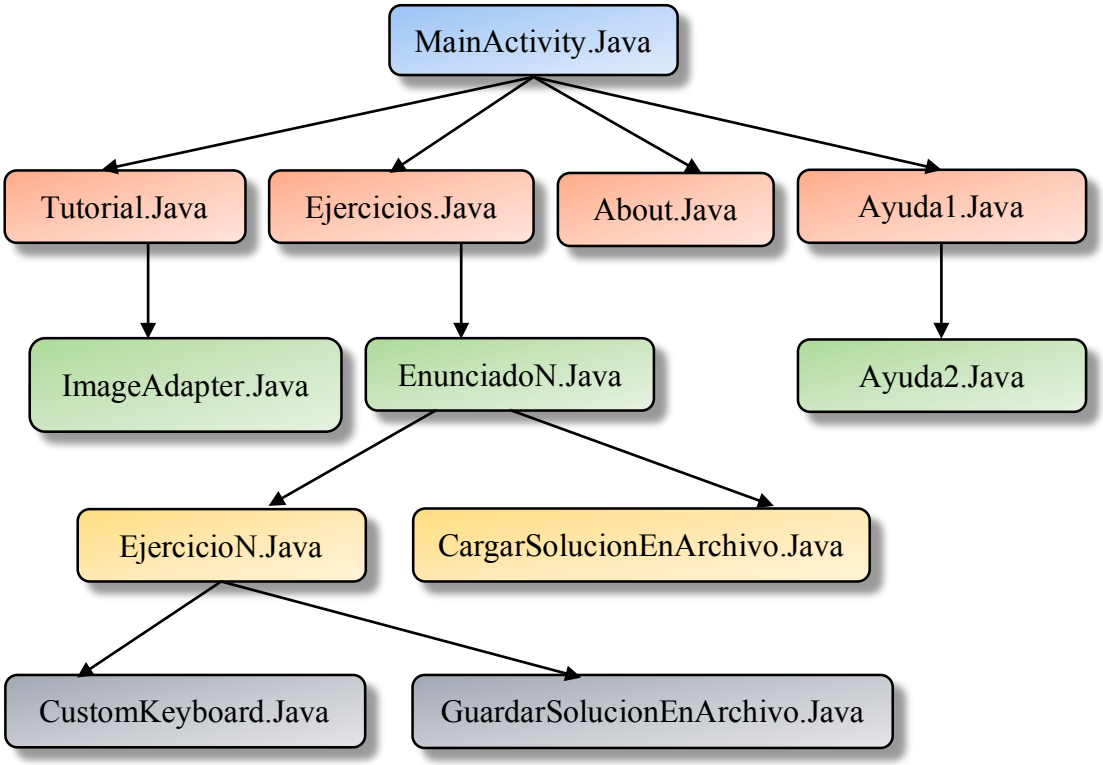


Figura 0-16: Diagrama de clases

Clase	Breve Descripción
About	Sirve para mostrar la sección "Acerca de" de la aplicación.
Ayuda1	Muestra las distintas clases de ayuda disponibles.
Ayuda2	Selecciona la ayuda deseada y ayuda a visualizarla.
CargarSolucionEnArchivo	Sirve para leer las soluciones guardadas de cada ejercicio y cargarla sobre el mismo.
CustomKeyboard	Define las acciones que podemos realizar con los dos teclados personalizados que se han creado.
Ejercicios	Sirve para seleccionar a que ejercicio de los 18 disponibles se quiere acceder
EnunciadoN	Sirve para seleccionar si se quiere iniciar un ejercicio. También se utiliza para acceder a la opción de carga o borrado de una solución.

EjercicioN	Esta clase se utiliza para la resolución de un determinado ejercicio
GuardarSolucionEnArchivo	A través de esta clase se gestionan los datos introducidos en los ejercicios para poder ser guardados en un archivo.
ImageAdapter	Es la clase utilizada para cargar todas las imágenes que serán utilizadas en el tutorial.
MainActivity	Es la clase utilizada para mostrar el menú principal de la aplicación. En ella se ha desarrollado la opción para enviar las soluciones por correo electrónico.
Tutorial	Sirve para la correcta visualización de las imágenes que se mostrarán en forma de diapositivas.

Tabla 0-1: Clases Java utilizadas

Método	Breve Descripción
onCreate()	Utilizado para realizar todas las acciones que deseemos que ocurran al inicio de una actividad.
afterTextChanged()	Utilizado para detectar cambie en los EditText.
onClick()	Realiza una acción al pulsar sobre algún objeto determinado.
onBackPressed()	Utilizado para gestión botón Atrás.
onKeyDown()	Cierra la actividad al pulsar el botón Atrás.
Guardar_TB()	Sirve para guardar estado de los ToggleButton
Save()	
Reset()	Resetea los ejercicios.
Ayuda()	Carga la solución en los ejercicios tipo tabla.
Solve()	Carga la solución en los ejercicios de formatos.
Comprobación()	Comprueba la solución introducida en el ejercicio
GuardarPreferencias()	Escribe los datos introducidos en el archivo de preferencias.
CargarPreferencias()	Carga los datos del archivo de preferencias en el ejercicio.
onCreateOptionsMenu()	Sirve para crear el menú emergente
onDraw()	Nos permite dibujar en la vista.
onTouchEvent()	Detecta donde toca el usuario en la pantalla.
lanzarEj()	Inicia actividad de ejercicios desde el enunciado.
isCustomKeyboardVisible()	Indica si el teclado personalizado es visible.
showCustomKeyboard()	Muestra teclado personalizado.
hideCustomKeyboard()	Oculta teclado personalizado.
registerEditText()	Activa el teclado personalizado en un EdiText determinado

Tabla 0-2: Principales métodos utilizados